2011

ALGORITHMS FOR CORRECTING NEXT GENERATION SEQUENCING ERRORS

Farideh Fazayeli

ALGORITHMS FOR CORRECTING NEXT GENERATION SEQUENCING ERRORS

(Spine title: Algorithms for Correcting Next Generation Sequencing Errors)
(Thesis format: Monograph)

by

Farideh Fazayeli

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....
Dr. Lucian Ilie

Joint Supervisor:

.....
Dr. Silvana Ilie

Examiners:

.....
Dr. Roberto Solis-Oba

.....
Dr. Sheng Yu

.....
Dr. Greg Gloor

The thesis by

Farideh Fazayeli

entitled:

Algorithms for Correcting Next Generation Sequencing Errors

is accepted in partial fulfillment of the
requirements for the degree of
Master of Science

.....
Date

.....
Chair of the Thesis Examination Board

Abstract

The advent of next generation sequencing technologies (NGS) generated a revolution in biological research. However, in order to use the data they produce, new computational tools are needed. Due to significantly shorter length of the reads and higher per-base error rate, more complicated approaches are employed and still critical problems, such as genome assembly, are not satisfactorily solved. We therefore focus our attention on improving the quality of the NGS data. More precisely, we address the error correction issue. The current methods for correcting errors are not very accurate. In addition, they do not adapt to the data. We proposed a novel tool, HiTEC, to correct errors in NGS data. HiTEC is based on the suffix array data structure accompanied by a statistical analysis. HiTEC's accuracy is significantly higher than all previous methods. In addition, it is the only tool with the ability of adjusting to the given data set. In addition, HiTEC is time and space efficient.

Keywords: Next generation sequencing, error correction, suffix array, statistical analysis.

Acknowlegements

I would like to express my sincere gratitude to Drs. Lucian Ilie and Silvana Ilie, for their continuous support of my M.Sc. study and research. Their guidance, encouragement, and immense knowledge helped me throughout my research and writing of this dissertation.

My special appreciation goes to my family for their endless love in my whole life. It would have been impossible for me to finish this work without their encouragement, understanding, support, and help.

I owe my loving thanks to my best friend, Hamed Kajbaf, for all the emotional support, camaraderie, enthusiasm, caring, and helping me to get through the difficult times.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

DNA sequencing, determining the order of the nucleotide bases in a molecule of DNA, is a challenging job since there is no machine that can sequence an entire molecule of DNA at a time. The technology is in fact very far from that. Very short pieces can be sequenced and computers are used to assemble them into longer sequences or use them in many other ways. The well known Sanger sequencing method [46] has been used for the sequencing of the first human genome [32, 61]. When it was introduced, the Sanger method improved very much the speed of the DNA sequencing process. However, the Sanger method is still very slow and very expensive. Therefore, the so-called next generation sequencing (NGS) technologies (or, high-throughput) have been developed, such as Roche's 454 sequencing, Illumina's Genome Analyzer (Solexa technology), or Applied Biosystems's SOLiD platform; see [36]. They can sequence billions of nucleotides in a single run. By comparison, the Sanger method produces half a million base-pairs per run. At the same time, the cost of NGS data is much lower.

Next-generation sequencing technologies have a variety of applications such as de-novo genome assembly [14, 17, 21, 24, 53], read mapping [10, 18, 25, 30, 33], full-genome re-sequencing, i.e., comprehensive polymorphism and mutation discovery in individual human genomes (detecting mutations or polymorphisms) [63], small RNA sequencing, i.e., microRNA profiling [39], chromatin immunoprecipitation-sequencing (ChIP-Seq), i.e., genome-wide mapping of protein-DNA interactions [26], etc.

However, there are several issues behind these novel technologies that make it difficult to use the generated data. The reads produced are significantly shorter, 35-100bp compared to 500-1000bp in the Sanger technology, and have higher per-base error rate. New methods and algorithms are required to deal with the huge amount of data produced by these novel technologies.

1.1 Problem statement

Among several fundamental computational problems concerning NGS data, genome assembly and read mapping have been investigated the most. The former attempts to align and merge the reads in order to reconstruct the genome that originated them [14, 17, 21, 24, 53]. The latter attempts to determine the location of newly sequenced reads against a reference genome of the same species [10, 18, 25, 30, 33].

Several issues such as huge amounts of data, repeats in the genome, and sequencing errors are common among all these novel tools. Whereas not much can be done about the first two, quite a bit of research has been done on finding methods for correcting errors [13, 14, 17, 50, 52, 58, 65].

The general idea for correcting reads is as follows. In order to determine the erroneous bases in the reads, the high coverage of the current sequencing technologies is used. Each base is usually sampled many times and the correct value will prevail. Approaches that work for Sanger reads, e.g., the algorithm of [58], based on multiple alignment, are too time consuming for the huge amount of data generated by NGS. New ideas are needed.

The SHARCGS assembly tool [17] filters erroneous reads before assembling of the genome. Reads are confirmed to be kept if they are generated multiple times and overlapping partners exist. However, this reduces the coverage. The version of the Euler assembler [45] for short reads, Euler-SR, [13, 14], includes its own method for correcting errors. The method is based on the spectral alignment algorithm [45]. Shi et al. proposed an efficient implementation of Euler-SR on a CUDA hardware [52]. They run the spectral alignment algorithm for correcting the errors in parallel. Their experiments show that the CUDA implementation is of $3 - 63$ times faster than the Euler-SR program. Schroder et al., [50], proposed a new method, called SHREC, for error correction based on weighted suffix trees. A weighted suffix tree of all reads and their reverse complement are constructed. If the weight of a node is less than a threshold, then the string labeling the node is suspected to be an error and correct otherwise. An erroneous node is corrected to one of its neighbor nodes. They showed their algorithm can outperform the Euler-SA algorithm but still lots of parameters are required to be tuned quite a bit to achieve a high accuracy. Simultaneously with our work, Yang et al. [65] proposed Reptile, also based on the k-spectrum approach of Euler-SR and CUDA.

1.2 Research objectives

All methods mentioned above often have modest accuracy. In addition, their parameters do not adapt to the data. This is a serious issue since, in real applications, testing of different parameters to see which perform the best is not possible. The goal of this thesis is to find a new, more accurate, method which also adapts to the data automatically.

We proposed a new method, HiTEC — High Throughput Error Correction — based on the suffix array data structure and accompanied by statistic analysis to tune parameters automatically. Our contribution in this work is as follows:

1. Show how to use statistics to find automatically the best parameters based on the input data.

2. Develop a C++ code based on the proposed method to identify and correct errors.

3. Test our software on real and simulated data and compare it with previous methods.

We evaluated the methods based on accuracy, that is, the ratio between the number of corrected reads and the number of initially erroneous reads. HiTEC's accuracy is significantly higher than the accuracy of all the other programs.

During testing, we ran all programs with default parameters. In some cases, that means their performance decreased significantly from that claimed by the paper. For instance, there is a significant difference between the accuracy obtained by running the SHREC program and that provided in [50].

For HiTEC, only the genome length and per-base error rate is required to be given as the input parameters. We evaluate our algorithm on a wide range of read lengths and coverage levels. Our experiments reveal that HiTEC is not only more accurate than previous works but also more robust. Its performance is affected very little by coverage level or read length.

In addition to accuracy, we evaluate HiTEC in terms of time and space complexities and showed that it is very efficient with respect to both. Our current serial implementation of HiTEC is comparable with Reptile and it is about six times faster than the parallel implementation of SHREC on the four-processor machine we used for testing. The space consumption is comparable with Reptile is and lower than that of SHREC for all tests. Nevertheless, we plan to improve the time and space complexity of our algorithm by providing a parallel implementation.

1.3 Thesis overview

In Chapter 2, a brief introduction on DNA Sequencing is given, containing a background on the Sanger method and next generation technologies, followed by their achievements and limitations. In Chapter 3, we present the most relevant related literature describing the latest methods developed for correcting errors in NGS data. Chapter 4 explains in depth the proposed approach. We have tested in Chapter 5 our algorithm on many data sets, simulated or real, from [50, 52, 65] as well as on several new ones. Finally, in Chapter 6, we present the conclusions and future work.

Chapter 2

DNA SEQUENCING

2.1 DNA

DNA, or deoxyribonucleic acid, is a macromolecule (polynucleotide) that contains the biological instructions required for an organism to develop, survive and reproduce. Its also referred to as a "genetic blueprint" since it contains the instructions required to construct other components of cells, such as proteins and RNA macromolecules which make each species unique. All information in the DNA is encoded by four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T), illustrated in Figure 2.1. Each nucleotide is composed of a sugar molecule (deoxyribose) bonded to a phosphate functional group and one of the four bases A, C, G, and T. The number of base pairs that constitute the DNA sequences of a few organisms is listed in Table 2.1. Human DNA consists of about 3 billion bases, and more than 99 percent of those bases are common in all people. The precise order of these bases determines the available information for an organism.

The exact structure of DNA was unknown until 1953 when the double-stranded helix model of DNA structure was proposed by James D. Watson and Francis Crick [62], which was essentially based on the x-ray data collected by Rosalind Franklin. The proposed double-stranded helix model initiated a revolution in molecular biology (Figure 2.2). Each strand has a direction and by convention is always read in the 5' end to 3' direction. The 5' end of DNA has a terminal phosphate (-PO4) group and the 3' end has a chemically different hydroxyl (-OH) terminal group. In the DNA double helix, one strand has an opposite direction versus the other strand. Each base within one strand binds with a complementary base in the other strand with A pairing up with T, and C with G. This arrangement of two nucleotides binding together across the

Organism	Number of base pair
Escherichia coli (bacterium)	4×10^6
Saccharomyces cerevisiae (yeast)	1.35×10^7
Drosophila melanogaster(insect)	1.65×10^8
Homo sapiens(human)	2.9×10^9
Zea mays(corn)	5.0×10^9

Table 2.1: Number of base pairs of DNA in different organisms.

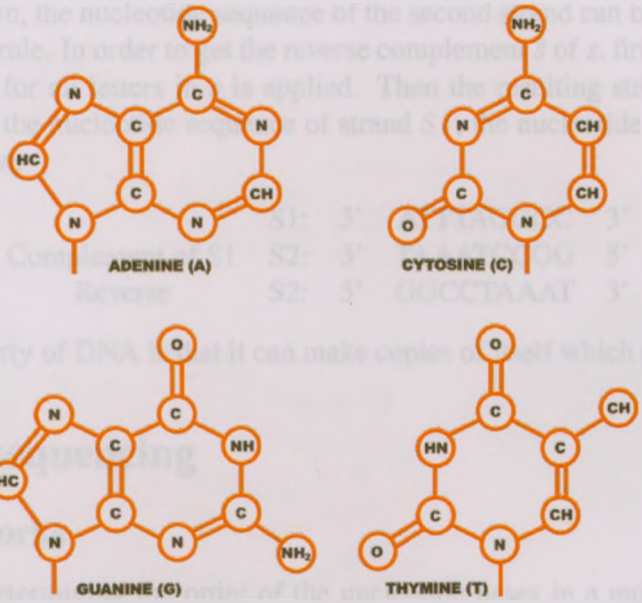


Figure 2.1: DNA Bases [2]

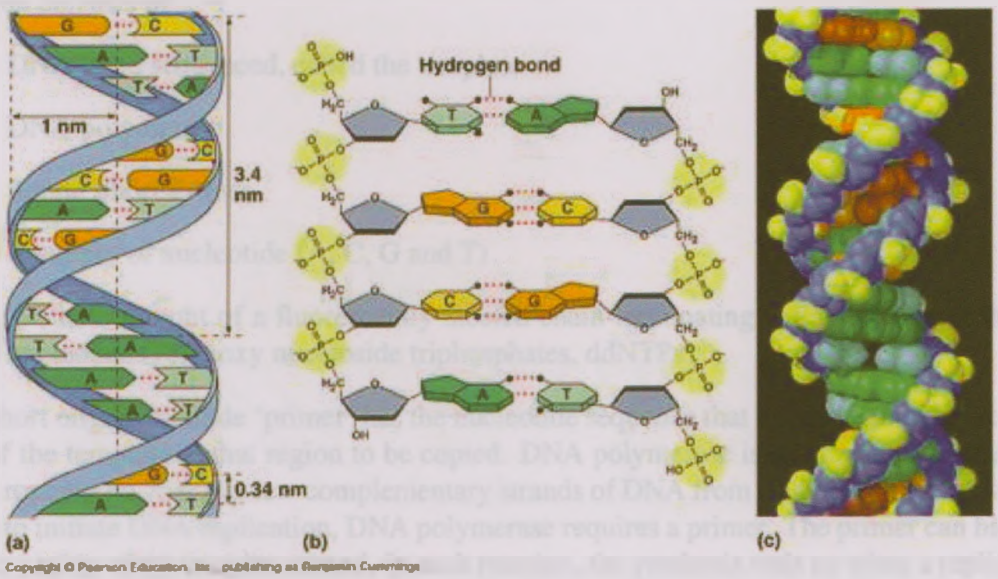


Figure 2.2: DNA Structure [1]

double helix, A-T and C-G, is known as complementary base pairing. Therefore, one strand of DNA is called the reverse complement or the complementary strand. If the nucleotide sequence of one strand is known, the nucleotide sequence of the second strand can be determined via the reverse complement rule. In order to get the reverse complement $\bar{s}$ of s, first the transformation A $\leftrightarrow$ T and C $\leftrightarrow$ G for all letters in s is applied. Then the resulting string is reversed. For example if we know the nucleotide sequence of strand $S1$, the nucleotide sequence of $S2$ can be identified as follow,

	S1:	5'	ATTTAGGCC	3'
Complement of S1	S2:	3'	TAAATCCGG	5'
Reverse	S2:	5'	GGCCTAAAT	3'

One important property of DNA is that it can make copies of itself which are called replicates.

2.2 Sanger sequencing

2.2.1 How it works

DNA sequencing, determining the order of the nucleotide bases in a molecule of DNA, is a challenging task since there is no machine that can sequence an entire long molecule of DNA at a time. In 1977, Sanger invented a novel method for DNA sequencing [46] for which he was awarded the Chemistry Nobel Prize in 1980. The basics of the Sanger Method are described below.

First, the DNA must be obtained in the single-stranded form and amplified. In order to sequence a piece of amplified DNA, four reaction mixtures are set up (Figure 2.3). Each reaction consists of

- DNA to be sequenced, called the template
- DNA polymerase
- Single Short Primer
- A supply of nucleotide (A, C, G and T)
- A small amount of a fluorescently labeled chain-terminating variant of one of the four nucleotides (dideoxy nucleoside triphosphates, ddNTPs).

The short oligonucleotide 'primer' has the nucleotide sequence that is complementary to the 3' end of the template at that region to be copied. DNA polymerase is an enzyme for catalyzing DNA replication, making new complementary strands of DNA from single-strand templates. In order to initiate DNA replication, DNA polymerase requires a primer. The primer can bind to a known region of the template strand. In each reaction, the synthesis ends up when a replication-terminating nucleotide is incorporated randomly into the growing DNA strand. The products of the reaction mixtures are then loaded into separate lanes of an electrophoresis gel. The presence of a ddNTP at the terminus can be recorded on a computer based on the different signal that is transmitted by ddNTP (Figure 2.4). Amplified DNA is then separated from smallest to largest

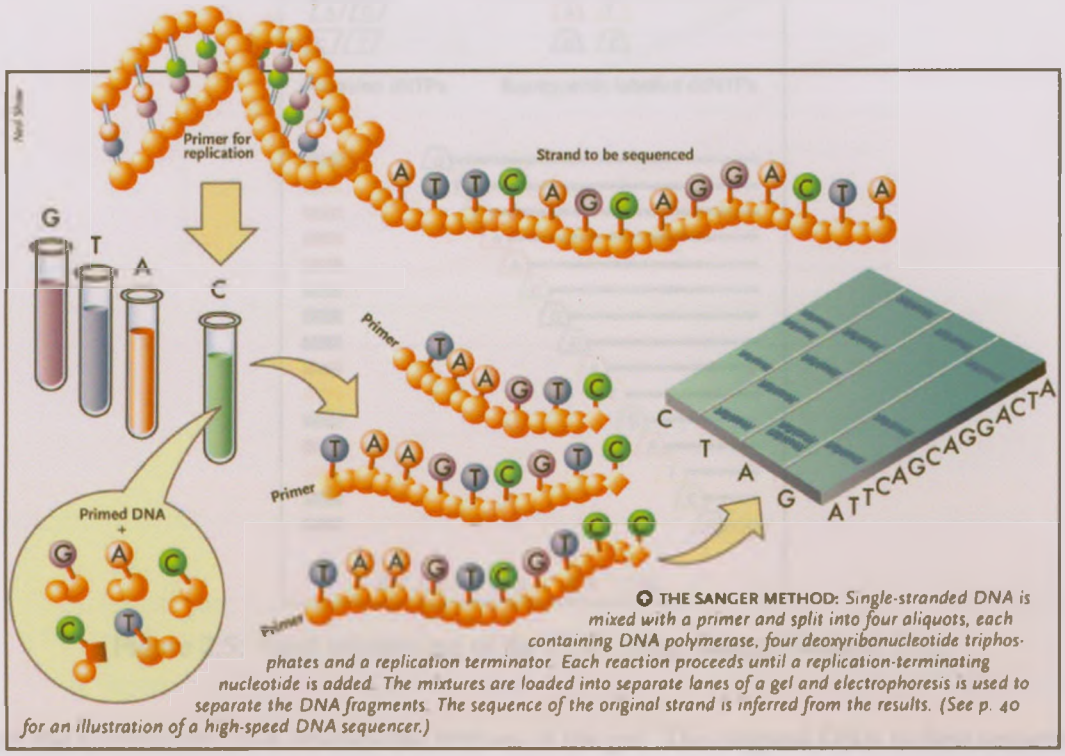


Figure 2.3: DNA sequencing using the Sanger method [3].

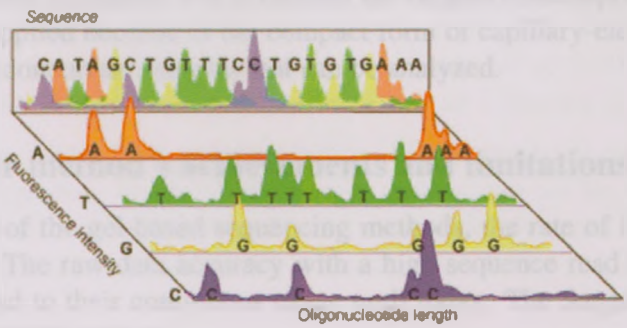


Figure 2.4: An electropherogram of a finished sequencing reaction using the Sanger method [20].

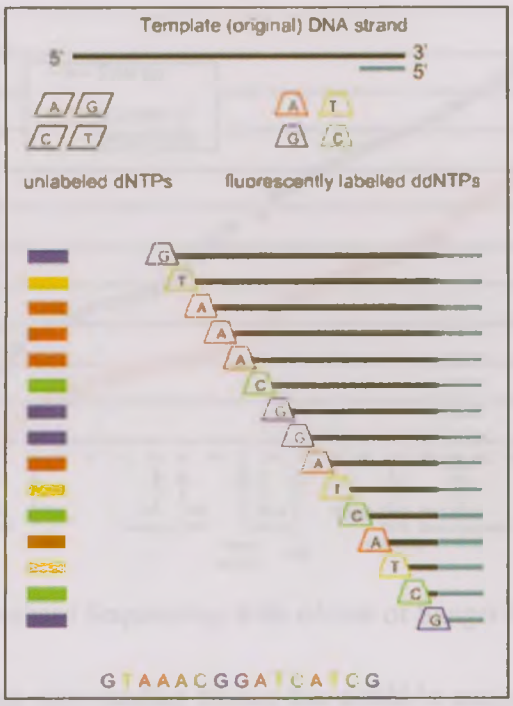


Figure 2.5: Final sequencing of the DNA using Sanger Method [43].

when the fluorescent DNA reaches the bottom of the gel. The original DNA is then sequenced by length and color of the truncated DNA products (Figure 2.5).

The original method was primarily a manual endeavor and hard to automate. Radioactive labeling of primer and using four separate reactions in Sanger's method made it non-user friendly and time consuming. Therefore, two modified versions of the first generation sequencing have been developed. In 1985, radioactive labeling was replaced with color fluorescent dyes [57]. It generates the possibility of mixing all four chain-termination reactions in one tube and thereby fastening the sequencing. In the second version [66], capillary-electrophoresis is used instead of slab-gel based separation which reduces the reagent consumption. Thereby, higher parallelism could be applied because of the compact form of capillary-electrophoresis that increases the number of concurrent samples that can be analyzed.

2.2.2 The Sanger method's achievements and limitations

With the introduction of the gel-based sequencing methods, the rate of DNA sequencing increased (Figure 2.6). The raw data accuracy with a high sequence read length generated by these technologies, lead to their continuous usage until today. The Sanger Method has had a variety of accomplishments such as:

- Complete sequencing of the phi X genome in 1977 [8, 56] which was a revelation in genome coding such as,
  - Translation of a DNA sequence in all possible reading frames for the first time.

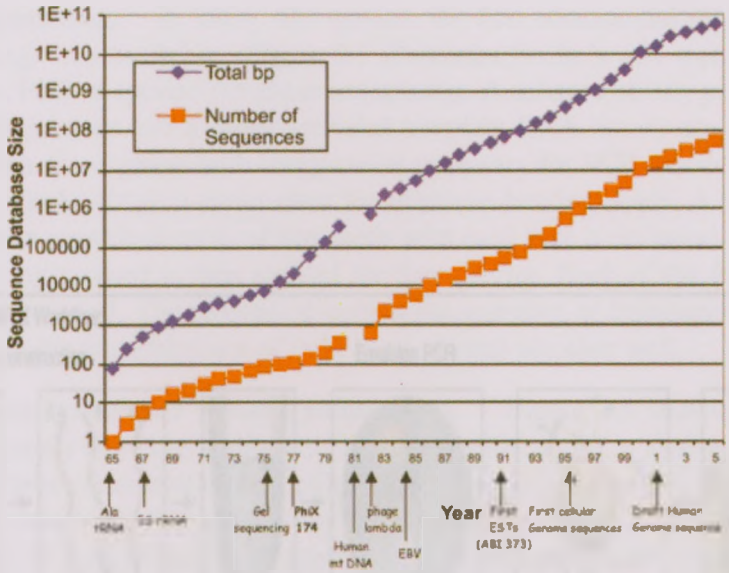


Figure 2.6: Growth of database Sequencing with advent of Sanger Sequencing Method [22]

  - Identifying long open reading frames that could be assigned to genes identified by traditional genetic methods.
  - Revealing that more than one reading frame were used to translate significant portions of a genome to produce two different proteins.

- Sequencing of the simian virus SV40 in 1978 [19]
- Sequence and organization of the 16.5 kb human mitochondrial genome in 1981 [4]
- Nucleotide sequence of the 48.5 kb complete bacteriophage lambda DNA in 1982 [47]
- Sequencing of the 172 kb EpsteinBarr virus in 1984 [6]
- Sequencing of the 237 kb human cytomegalovirus genome in 1991 [7]
- Complete Sequencing of first Human Genome in 2001 [32, 61]. The project started in 1990 with an expected time frame of 15 years and cost of 5-10 billion USD. It was successfully finished in 13 years with an actual cost of 3 billion USD.

Reducing cost of sequencing via miniaturization, higher degree of parallelism, and faster analysis are highly demanded. However, the Sanger method can not be improved further because it relys on electrophoretic separation. Therefore, completely novel methods of sequence generation were required to overcome all above limitations.

2.3 Next Generation Sequencing (NGS)

The Sanger method is very slow and very expensive per base. Therefore, the so-called next generation sequencing (NGS) technologies (or, high-throughput) have been developed. Different platforms have been developed for NGS as follows [51].

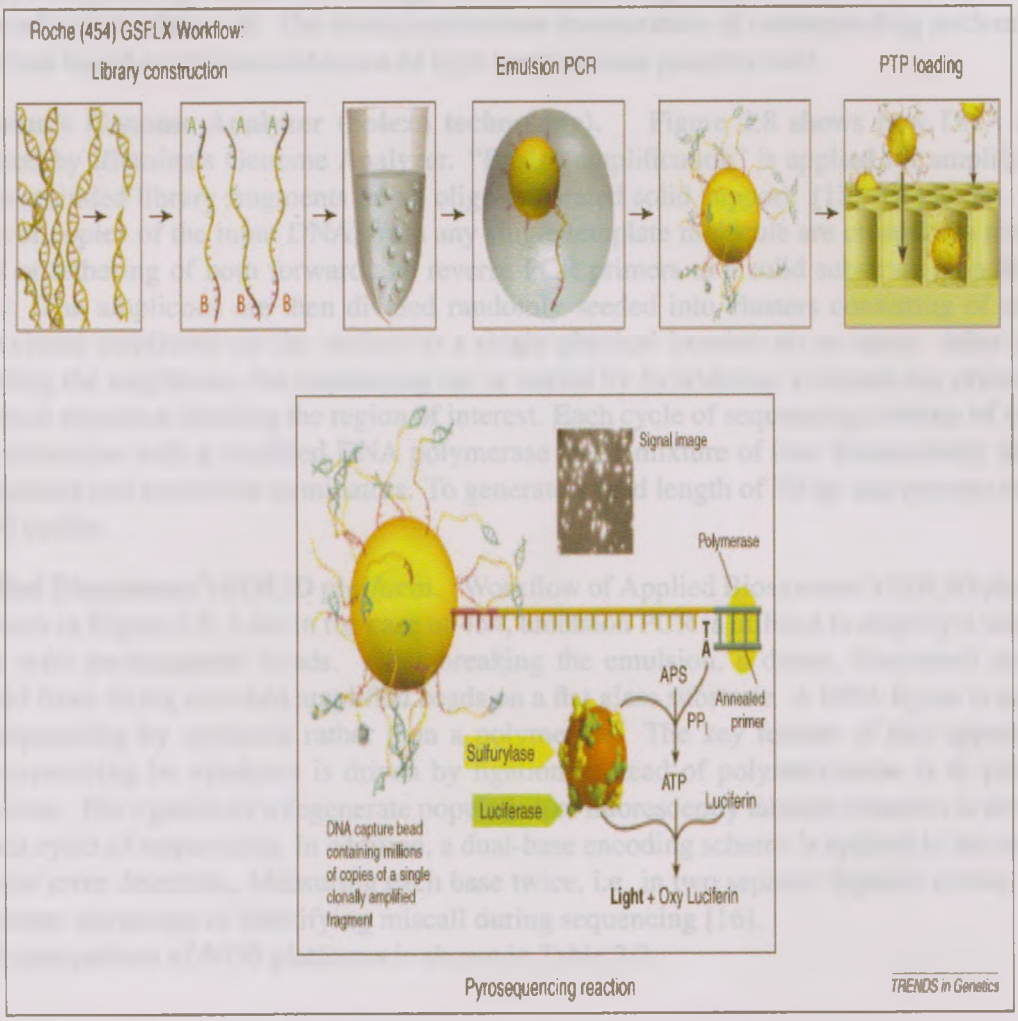


Figure 2.7: DNA Sequencing with Roche's 454 [36]

Roche's 454 sequencing. In 2004, 454 system, the first commercial product of next generation sequencing, was available. Figure 2.7 illustrates Roche's 454 workflow. In this approach, emulsion PCR is applied for clone sequencing. A mixture of tiny paramagnetic beads coated with DNA primers and a single-stranded template DNA library are prepared in aqueous droplets within an oil phase with components necessary for PCR reaction. Beads are then enriched based on hybridization enrichment for amplicon-bearing beads. A picotiter plate fabricated is utilized in organized array of tiny wells with each hole is occupied by only one bead. The pyrosequencing method is then applied for Sequencing. Each of the four dNTPs are introduced into the flow cell. The incorporation/non-incoporation of corresponding nucleotide is identified based on presence/absence of light burst of each picotiter well.

Illumina's Genome Analyzer (Solexa technology). Figure 2.8 shows how DNA is sequenced by Illumina's Genome Analyzer. "Bridge amplification" is applied for amplifying a single-stranded library fragments on an oligo-decorated solid support. [13] Amplicons, thousands of copies of the input DNA, from any single template molecule are created via multiple times of tethering of both forward and reverse PCR primers to a solid substrate by a flexible linker. The amplicons are then divided randomly seeded into clusters consisting of around 1000 clonal amplicons on the surface to a single physical location on an array. After single stranding the amplicons, the sequencing run is started by hybridizing a sequencing primer to a universal sequence flanking the region of interest. Each cycle of sequencing consists of single-base extension with a modified DNA polymerase and a mixture of four fluorescently labeled nucleotides and reversible terminators. To generate a read length of 50 bp this process repeats for 50 cycles.

Applied Biosystems's SOLiD platform. Workflow of Applied Biosystems's SOLiD platform is shown in Figure 2.9. Like in the case of 454, emulsion PCR is utilized to amplify a template DNA with paramagnetic beads. After breaking the emulsion, a dense, disordered array is created from fixing enriched amplified beads on a flat glass substrate. A DNA ligase is applied for sequencing by synthesis rather than a polymerase. The key feature of this approach is that sequencing by synthesis is driven by ligation, instead of polymerization as in previous platforms. The ligation of a degenerate population of fluorescently labeled octamers is involved in each cycle of sequencing. In addition, a dual-base encoding scheme is applied in the process to assist error detection. Measuring each base twice, i.e. in two separate ligation cycles, leads to another advantage of identifying miscall during sequencing [16].

A comparison of NGS platforms is shown in Table 2.2.

2.4 Applications of NGS

Next-generation sequencing technologies have been applied to a variety of applications [51] such as

- Genome assembly which attempts to align and merge the reads in order to reconstruct the genome that originated them [14, 17, 21, 24, 53].
- Reads mapping which attempts to determine the location of newly sequenced reads against a known different reference genome of the same species [10, 18, 25, 30, 33].

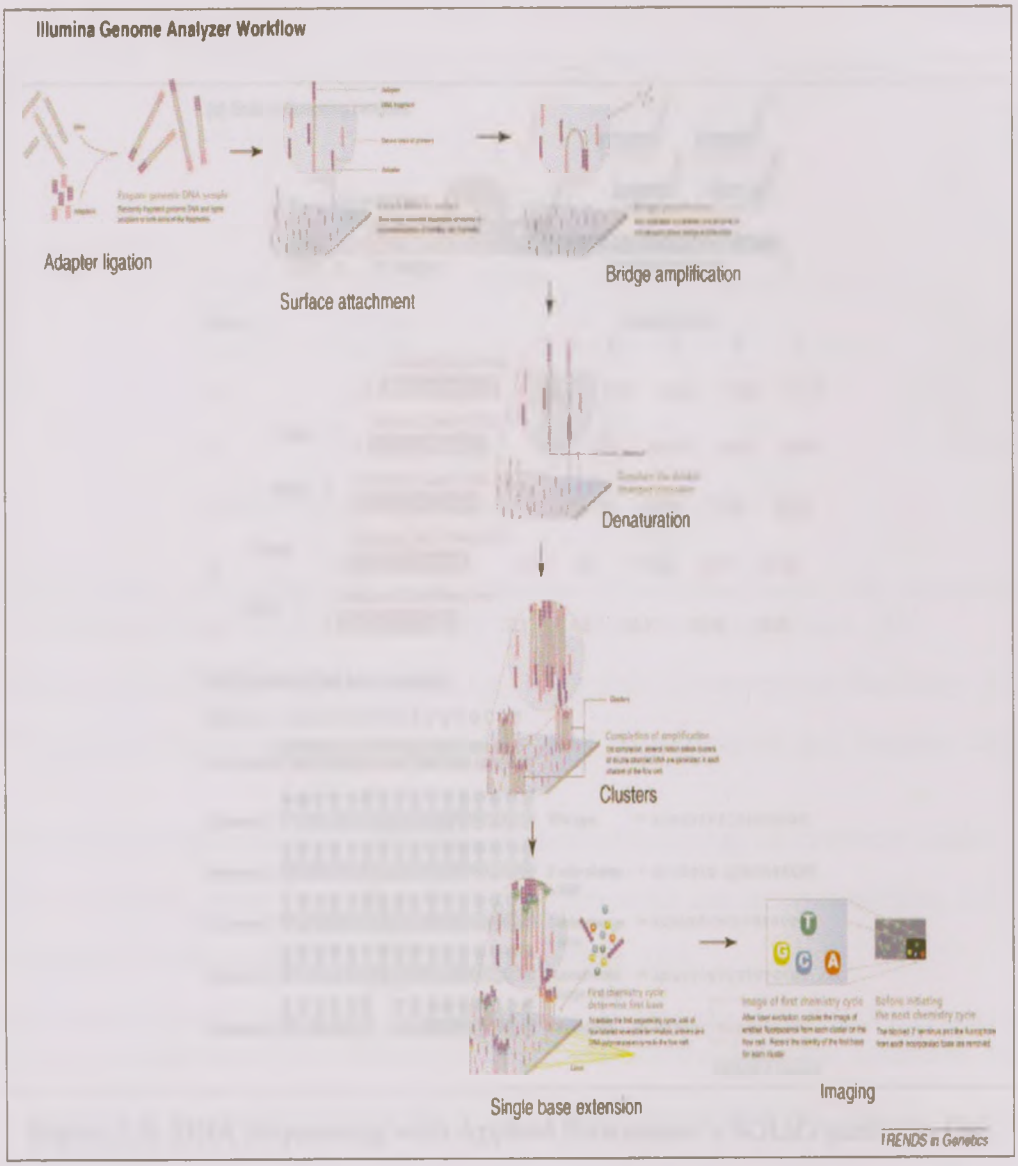


Figure 2.8: DNA Sequencing with Illumina's Genome Analyzer (Solexa technology) [36]

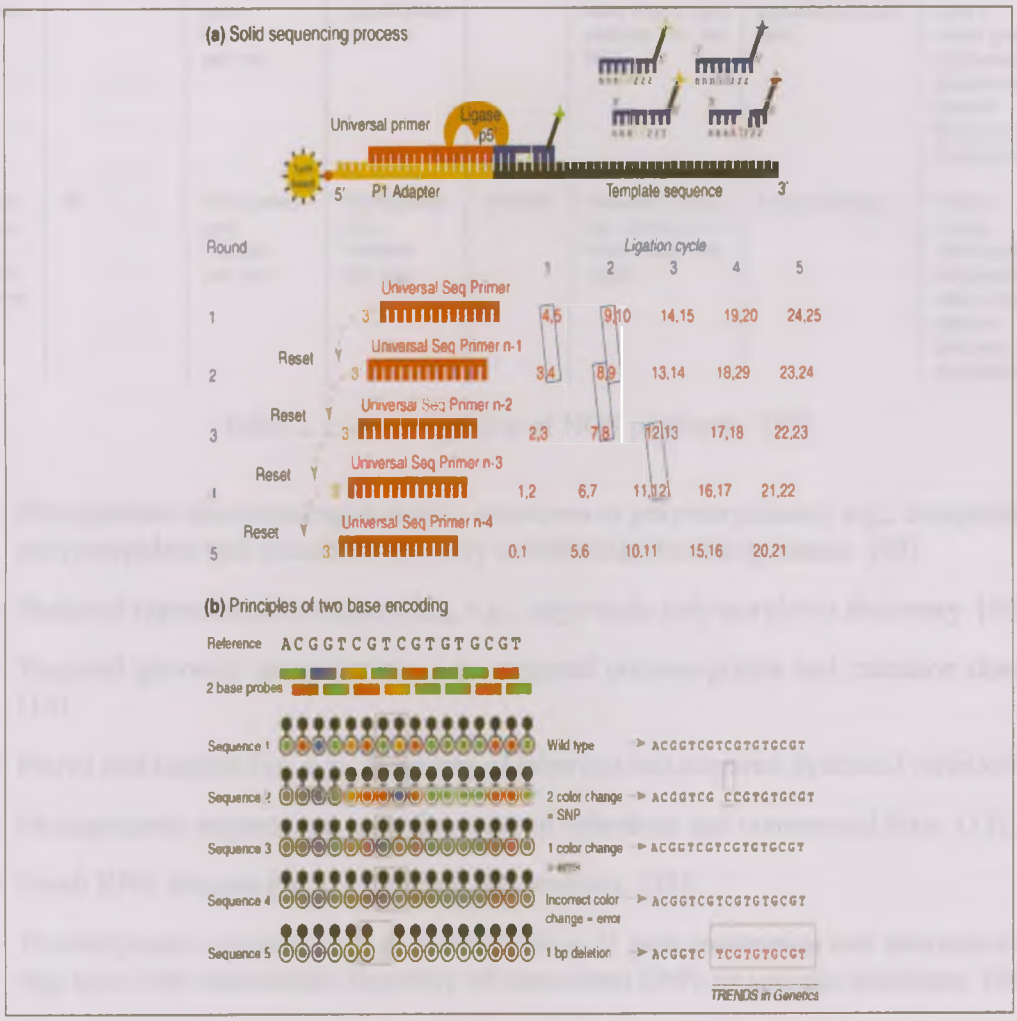


Figure 2.9: DNA Sequencing with Applied Biosystems's SOLiD platform [36]

Platform	Read lengths(bases)	Sequencing run time(days)	Total bases per run (Gb)	Machine cost (USD)	Pros	cons	Biological Applications
Roche's 454	330(Average)	0.35	0.45	500,000	Longer reads improve mapping in repetitive regions; fast run times	High reagent cost; high error rates in homopolymer repeats	Bacterial and insect genome de novo assemblies; medium scale (< 3 Mb) exome capture; 16S in metagenomics
Illumina's Genome Analyzer	75 or 100	4(Fragment run), 9(Mate-pair run)	18(Fragment run),35(Mate-pair run)	540,000	Currently the most widely used platform in the field	Low multiplexing capability of samples	Variant discovery by whole-genome resequencing or whole-exome capture; gene discovery in metagenomics
Applied Biosystems's SOLiD platform	50	7(Fragment run), 14(Mate-pair run)	30(Fragment run), 50(Mate-pair run)	595,000	Two-base encoding provides inherent error correction	Long run times	Variant discovery by whole-genome resequencing or whole-exome capture; gene discovery in metagenomics

Table 2.2: A comparison of NGS platforms [37]

- Full-genome resequencing(detecting mutations or polymorphisms), e.g., comprehensive polymorphism and mutation discovery in individual human genomes [63].
- Reduced representation sequencing, e.g., large-scale polymorphism discovery [60].
- Targeted genomic resequencing, e.g., targeted polymorphism and mutation discovery [16].
- Paired end sequencing, e.g., discovery of inherited and acquired structural variation [11].
- Metagenomic sequencing, e.g., discovery of infectious and commensal flora [15].
- Small RNA sequencing, e.g., microRNA profiling [39].
- Transcriptome sequencing, e.g., quantification of gene expression and alternative splicing; transcript annotation; discovery of transcribed SNPs or somatic mutations [64].
- Sequencing of bisulfite-treated DNA, e.g., determining patterns of cytosine methylation in genomic DNA [34].
- Nuclease fragmentation and sequencing, e.g., nucleosome positioning [49].
- Molecular barcoding, e.g., multiplex sequencing of samples from multiple [41].
- Chromatin immunoprecipitation sequencing (ChIP-Seq), e.g., genome-wide mapping of protein-DNA interactions [26].

Several software programs available for analyzing NGS data are listed in Table 2.3.

Program	Category	Author(s)	Summary	URL
ELAND	Alignment	Anthony J. Cox	Efficient Large-Scale Alignment of Nucleotide Databases. Whole genome alignments to a reference genome.	*http://www.illumina.com/*
Exonerate	Alignment, Mapping	Guy S. Slater et al.	Various forms of alignment (including Smith-Waterman-Gotoh) of DNA/protein against a reference.	*http://www.ebi.ac.uk/~guy/exonerate*
MAQ	Alignment, variant detection	Heng Li	Mapping and Assembly with Qualities (renamed from MAPASS2)	*http://maq.sourceforge.net*
Mosaik	Alignment	Michael Stromberg et al.	Producing gapped alignments using the Smith-Waterman algorithm. Support for Roche's 454, Illumina, and SOLiD	*http://bioinformatics.bc.edu/marthlab/Mosaik*
RMAP	Alignment	Andrew Smith et al.	Assembles 20 - 64 bp Solexa reads to a FASTA reference genome.	*http://rulai.cshl.edu/rmap*
SHRiMP	Alignment	Michael Brudno et al.	Assembles to a reference sequence. Developed with Applied Biosystem's colourspace genomic representation in mind.	*http://compbio.cs.toronto.edu/shrimp*
SOAP	Alignment, variant detection	Ruiqiang Li et al.	SOAP (Short Oligonucleotide Alignment Program) is a program for efficient gapped and ungapped alignment of short oligonucleotides onto reference sequences. SOAP2 is an updated program based on Burrows-Wheeler Transform	*http://soap.genomics.org.cn*
SSAHA2	Alignment	Zemin Ning et al.	a pairwise sequence alignment program designed for the efficient mapping of sequencing reads onto genomic reference sequences	*http://www.sanger.ac.uk/Software/analysis/SSAHA2*
SXOligoSearch	Alignment	Synamatix	align Illumina reads against a range of Refseq RNA or NCBI genome builds for a number of organisms	*http://www.bioinformatics.org/wiki/SXOligoSearch*
ALLPATHS	Assembly	Jonathan Butler et al.	De novo assembly of whole-genome shotgun microreads.	*ftp://ftp.broadinstitute.org/pub/crd/ALLPATHS/*
Edena	Assembly	David Hernandez et al.	An assembler dedicated to process the millions of very short reads produced by the Illumina Genome Analyzer. Edena is based on the traditional overlap layout paradigm.	*http://www.genomic.ch/edena*
Euler-SR	Assembly	Mark Chaisson et al.	Contrary to the overlap-layout approach, EULER-SR uses a de Bruijn graph to construct an assembly.	*http://euler-assembler.ucsd.edu/portal/*
SHARCGS	Assembly	Juliane Dohm et al.		*http://sharcgs.molgen.mpg.de*
SHRAP	Assembly	Andreas Sundquist et al.	A sequencing protocol and assembly methodology that utilizes high-throughput short-read technologies.	
SSAKE	Assembly	Rene Warren et al.	A genomics application for aggressively assembling millions of short nucleotide sequences by progressively searching for perfect 3'-most k-mers using a DNA prefix tree.	*http://www.bcgsc.ca/platform/bioinfo/software/ssake*
vCAKE	Assembly	William Jeck	De novo assembly of short reads with robust error correction. An improvement on early versions of SSAKE.	*http://sourceforge.net/projects/vcake*
velvet	Assembly	Daniel Zerbino aet al.	A de novo genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454. Need about 20-25X coverage and paired reads.	*http://www.ebi.ac.uk/%7Ezerbino/velvet*
PyroBayes	Base caller	Aaron Quinlan et al.	It was designed to assign more accurate base quality estimates to the 454 pyrosequences.	*http://bioinformatics.bc.edu/marthlab/PyroBayes*
PbShort	variant detection	Gabor Marth		*http://bioinformatics.bc.edu/marthlab/PbShort*
ssahaSNP	variant detection	Zemin Ning et al.	It detects homozygous SNPs and indels by aligning shotgun reads to the finished genome sequence. Highly repetitive elements are filtered out by ignoring those kmer words with high occurrence numbers.	*http://www.sanger.ac.uk/Software/analysis/ssahaSNP*

Table 2.3: Bioinformatics tools for short-read sequencing [51].

2.5 Problems with NGS

Among several fundamental computational problems concerning NGS data, genome assembly and read mapping have been investigated the most. Several issues such as the huge amount of data, significantly short reads, repeats in genome, and sequencing errors are common among these different novel tools. The reads produced are significantly shorter, 35-100bp compared to 500-1000bp in Sanger technology and have higher per-base error rate. Since the read length are significantly short, large portions of a read set can not uniquely align and make assembly and alignment in mapping more difficult for NGS than for the Sanger method. In the following, we explain two other issues, repeats in the genome and sequencing errors.

Repeats

A segments of DNA repeated multiple times in the genome is called a repeat. Repetitive sequences are categorized into five classes based on their origin [32]:

1. Transposon-derived repeats (100-6,000 bp in length) also known as interspersed repeats;
2. Processed pseudogenes, that is, inactive retroposed copies of transcribed coding genes;
3. Low copy repeats derived from segmental duplications (10 kb-300 kb in length), that is, chunks of DNA copied from one region of the genome to another;
4. Microsatellites, that is, simple sequence tandem repeats, e.g., AAAAAAA, TATATATATA, or CGCCGCCGCCGCCGCCGC;
5. Minisatellites, which are blocks of tandemly repeated sequences (10-100 base in length).

Unlike the first three class of repeats that can be present at different locations across a whole genome, the last two classes known as tandem repeats occur consecutively in the genome. Tandem repeats fall into two sub-categories, primitive and non-primitive. If a tandem repeat does not contain other tandem repeats, it is called a primitive tandem repeat. For example, strings *aa* and *abab* are primitive tandem repeats, while *aaaa* is not a primitive tandem repeat. Detection of repeats is a well-studied problem in computational biology.

Tandem repeats play a role in regulation of gene expression. They are used as markers in mapping and population studies because of their higher rate of variation. In spite of the useful functions of repeats, these elements make assembly of shotgun fragments very difficult for complex (repeat-rich) genomes.

Gaps in assembly of genome can be produced because of missing some repeats. Repeats can produce false overlaps that make them to be collapsed and generate smaller number of copies which inaccurately sequenced. Repeats can confuse an assembler to misjoin nonadjacent genomic fragments together and generate false overlaps as illustrated in Figure 2.10.

Higher error rate

Reads sequenced by NGS have a higher error rate than for Sanger sequencing. Most of fragment assemblers rely on the overlap-layout-consensus paradigm which consists of three phases.

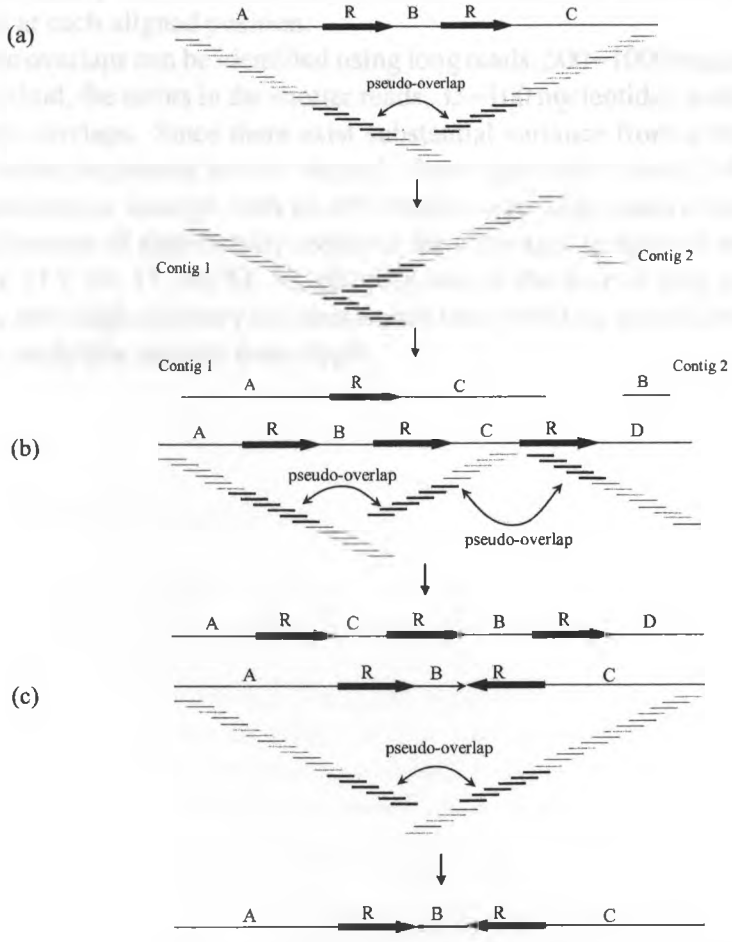


Figure 2.10: How repeats cause false overlapping in an assembler: (a) Collapsing two repeat copies (R) in the correct sequence (top) into one copy in the incorrect assembled sequence (bottom); (b) Flanking two DNA segments (B and C) by three repeat copies (R); (c) a DNA segment (B) flanked by two inverted repeat copies (R); orientation of B is changed in the misassembled genome sequence [59].

In the first phase, the overlaps between all possible pairs of reads are determined. In order to find the overlap between two reads, they are aligned in order to determine the overlap with the maximum score which is the sum of scores of each match, each mismatch, and each gap. In the second phase, the layout step, all reads are aligned together based on the overlaps between them. A group of overlapping reads that are ordered is called a contig which is oriented with respect to a region of the target genome. A list of ordered contigs oriented with respect to the target genome is called a scafold. In the last phase, the consensus sequence is built by taking major nucleotide at each aligned position.

While reliable overlaps can be identified using long reads, 500–1000 nucleotides, generated by the Sanger method, the errors in the shorter reads, 35–100 nucleotides, make it more difficult to determine firm overlaps. Since there exist substantial variance from a reference, coverage gaps can occur when sequences are not aligned. Some assemblers simply discard reads with ambiguous alignments, or contigs, with no information regarding relative order. However, this is not desirable because of significantly reducing the coverage. In spite of several methods of correcting errors [13, 14, 17, 50, 52, 58, 65] proposed in the several past years a method for correcting errors with high accuracy in a reasonable time with less parameters is still required. In this thesis we study this issue in more depth.

Chapter 3

NGS ERROR CORRECTING

Error correction methods applicable to Sanger reads are not suitable for NGS reads because of significanlty shorter read length and huge amount of data genereted by NGS technologies. Therefore, new tools have been developed. We review in this section the most important read correcting methods that have been developed for NGS data.

The genomes that were used in [50, 52] for comparing these methods are shown in Table 3.1.

3.1 Spectral alignment

The version of the Euler assembler [45] for short reads, Euler-SR, [13, 14], includes its own method for correcting errors. The method is based on the spectral alignment algorithm [45] which is described below. Let $R = \{r_1, r_2, \cdots r_n\}$ be a set of reads. Let s be a string over the alphabet Σ = {A, C, G, T}. A substring of s is any consecutive sequence of letters from s, i.e., $s[i \cdots j] = s[i]s[i+1] \cdots s[j]$; in particular $s = s[1 \cdots |s|]$. Each read r_i is a substring of length l. Let k-mer be any substring of the genome of length k. Assume two parameters k and m are given, then spectral alignment problem can be defined as follows.

Solid k-mer: A k-mer a is called solid with respect to m and R if there exist at least m reads in R with a substring a.

Reference genome (ID)	Accession no.	Len.(bp)
Saccharomyces Cerevisiae, Chr. 5 (S.cer5)	NC_001137	576,869
Saccharomyces Cerevisiae, Chr. 7 (S.cer7)	NC_001139	1,090,946
Haemophilus Influenzae (H.inf)	NC_007146	1,914,490
Escherichia coli str.K-12 substr.MG1655 (E.coli)	NC_000913	4,639,675
Escherichia coli str.K-12 substr.DH10B (E.coli2)	NC_010473	4,686,137
Staphylococcus aureus (S.aureus)	NC_003923	2,820,462
Helicobacter acinonychis (H.acinonychis)	NC_008229	1,553,927

Table 3.1: List of Genomes used for comparison. They can be downloaded from *http://www.ncbi.nlm.nih.gov/ using their accession number listed in the second column, Accession no.*

Weak k-mer: A k-mer which is not solid.

Spectrum: The set of all solid k-mers with respect to m and R is called the spectrum of the reads with respect to k, m, and R denoted by $T_{k,m}(R)$

Spectral Alignment Problem: Given a string s and a spectrum $T_{k,m}(R)$, determine the minimum number of mutations in s so as to minimize their distance to the spectrum T.

Pe'er and Shamir considered the same problem in a different context of resequencing by hybridization [44]. In [12], a dynamic approach for correcting errors is applied to Roche's 454 reads. It is based on a spectral alignment algorithm using edit distance as a distance function. In case of small number of mutations, a dynamic programming method can solve the spectral alignment problem efficiently even for large k. In [13, 14] an iterative approach is proposed to correct errors in Illumina reads. Since there is a few insertion/deletion in Illumina reads, substitutions (Hamming distance) can be considered as mutations (distance function). In their greedy approach, spectral alignments is called iteratively with the set of all reads and all solid k-tuples. In each iteration, the spectral alignment algorithm may change the sets of weak and solid k-mers in order to decrease the number of weak k-mers and increase the number of solid k-mers. A mutation is selected if the number of changed weak k-mers is bigger than a threshold t. The heuristic is called iteratively until there is no mutation that can change at least t weak k-mers or all k-mers are solidified.

3.2 SHREC

Schroder et al., [50], proposed a new method, called SHREC, for error correction based on weighted suffix trees. A brief introduction to suffix trees is given first.

3.2.1 Suffix Trees

Let s be a finite string over the alphabet $\Sigma = \{A, C, G, T\}$. Σ^* is the set of all strings over Σ and $|s|$ denotes length of string s. Let \$ be a termination character which does not belong to Σ. We change s by appending \$ at the end so that each suffix of s is unique. The suffix of s starting at the ith position is defined by $\text{suff}_i = s[i \cdots |s|] = s[i]s[i+1] \cdots s[|s|]$. A suffix tree ST of s is a tree with n leaves with labels $1, 2, \cdots, n$ where n. The concatenation of edges from the root to a leaf of the suffix tree labeled i is suff_i. Figure 3.1 shows a suffix tree generated from the string "CATTATTAGGA".

Path-label(x) is a the string obtained by concatenating the edge's strings on the path from the root to a node x. A weighted suffix tree is a generalized suffix tree where each node has a weight. The weight of node x represents the number of occurrences of the substring path-label(x) in string s.

3.2.2 SHREC Algorithm

A weighted suffix tree of all reads and their reverse complements (R) is constructed. The structure of the suffix tree in the presence of errors is illustrated in Figure 3.2. An error at

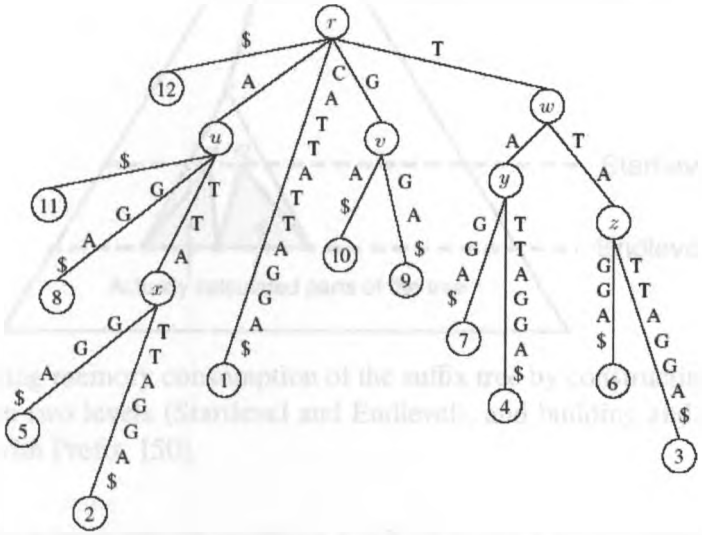


Figure 3.1: Suffix tree of the string "CATTATTAGGA" [5].

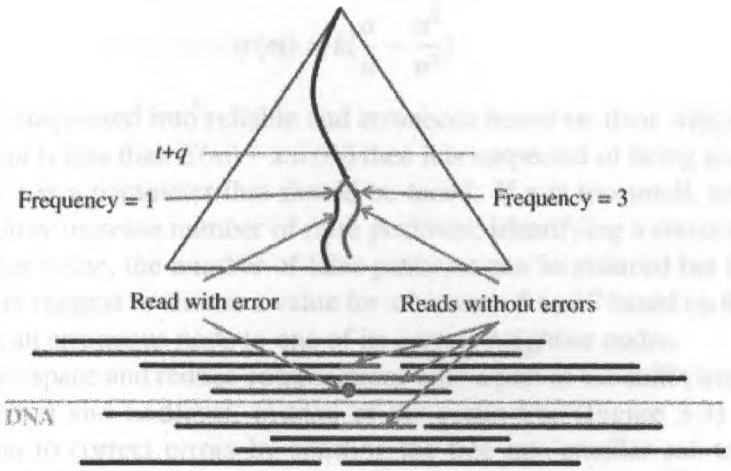


Figure 3.2: Changing the suffix tree structure in the presence of errors [50].

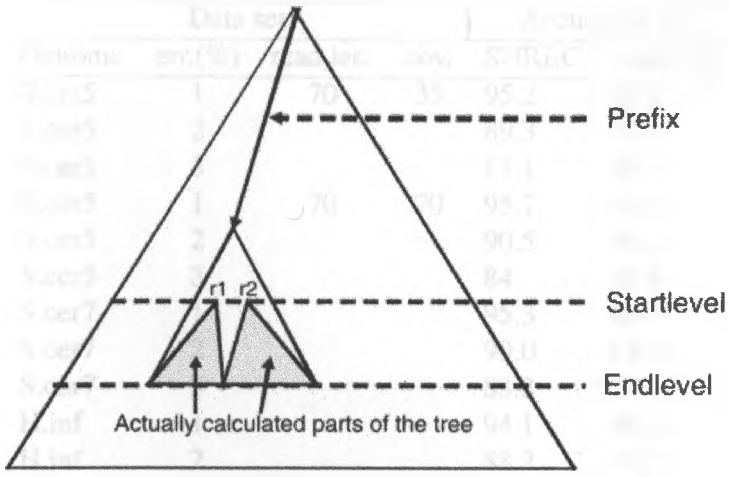


Figure 3.3: Reducing memory consumption of the suffix tree by constructing only a part of the suffix tree between two levels (Startlevel and Endlevel), and building and analyzing only the sub-tree starting with Prefix [50].

position k of a read with no error in positions 1 to $k-1$ causes a node at the corresponding level to have two children. The children nodes representing erroneous bases have smaller weight than expected.

The expected weight of a node depends on its level and the coverage of the genome. The expected weight of a node at level m is calculated based on the expectated number of occurrences of a substring of length m in R, (3.1), and the standard deviation value $\sigma(m)$, (3.2).

$$a = l - m + 1$$
$$E(m) = \frac{ka}{n} \tag{3.1}$$
$$\sigma(m) = k(\frac{a}{n} - \frac{a^2}{n^2}) \tag{3.2}$$

Then, nodes are categorized into reliable and erroneous based on their weights. If the weight of a node at level m is less than $E(m) - x.\sigma(m)$ then it is suspected of being an error and correct otherwise, where x is a parameter that should be tuned. If x is too small, more errors can be detected but this may increase number of false positives, identifying a correct base as an error. If it is set to a large value, the number of false positives can be reduced but it may miss some errors. The authors suggest to choose a value for x between 5 and 7 based on their experiments. They will correct an erroneous node to one of its correct neighbor nodes.

In order to save space and reduce computations only a part of the suffix tree is built between two levels, Startlevel and Endlevel, instead of the entire tree (Figure 3.3). They proposed a parallel program to correct errors by splitting the tree into smaller sub-trees (Figure 3.3). Each sub-tree starts with a prefix substring. Since correcting a node in a sub-tree is totally independent from another sub-tree, they run the correcting program in parallel on the sub-trees.

They showed their algorithm can outperform Euler-SA algorithm (Table 3.2) on a real and several simulated data sets from some of genomes listed in Table 3.1. Still many parameters must be tuned emparically to achieve high accuracy.

Data set				Accuracy (%)	
Genome	err.(%)	read len.	cov.	SHREC	Euler-SR
S.cer5	1	70	35	95.2	83.2
S.cer5	2			89.3	71.1
S.cer5	3			81.1	57.4
S.cer5	1	70	70	95.7	80.2
S.cer5	2			90.5	68.0
S.cer5	3			84	16.9
S.cer7	1			95.3	80.3
S.cer7	2			90.0	68.0
S.cer7	3			83.3	13.7
H.inf	1			94.1	80.0
H.inf	2			88.2	67.7
H.inf	3			81.0	53.5
E.coli	1			93.5	80.0
E.coli	2			87.4	67.7
E.coli	3			80.0	54.4
S.aureus	1	35	43	88.3	33.4

Table 3.2: Accuracy, percentage of corrected reads relative to the total number of erroneous reads, comparison between SHREC and Euler-SR [50].

3.3 CUDA implementation

H. Shi et al. proposed an efficient implementation of the error correcting algorithm of Euler-SR on CUDA hardware [52], mainly to improve the speed.

3.3.1 CUDA Programming model

Compute Unified Device Architecture (CUDA) is a parallel computing architecture developed by NVIDIA. CUDA processors are programmed in CUDA C, which is C/C++ with a CUDA extension (minimum extension of C) to write scalable multithreaded programs for CUDA-enabled GPUs [42]. The hardware model of CUDA is illustrated in Figure 3.4. The kernel is a sequential part of all CUDA programs representing the operations that can be performed by a single thread. A set of concurrent threads is called a thread block and a set of independent blocks is a grid.

Several kinds of memory are available in a CUDA architecture. Each thread has access to a local memory of size 16KB which is readable and writable and a set of readable and writable per-thread registers which is the fastest memory. All threads in a block have access to a readable and writable shared memory of size 16KB called per-block shared memory. Threads of different blocks can not communicate directly. The whole device (the GPU) has a readable and global memory of size around 1GB with high latency and low bandwidth. Kernels can read from a large cached texture memory using a texture fetching device function which is faster than reading from the global or local memory.

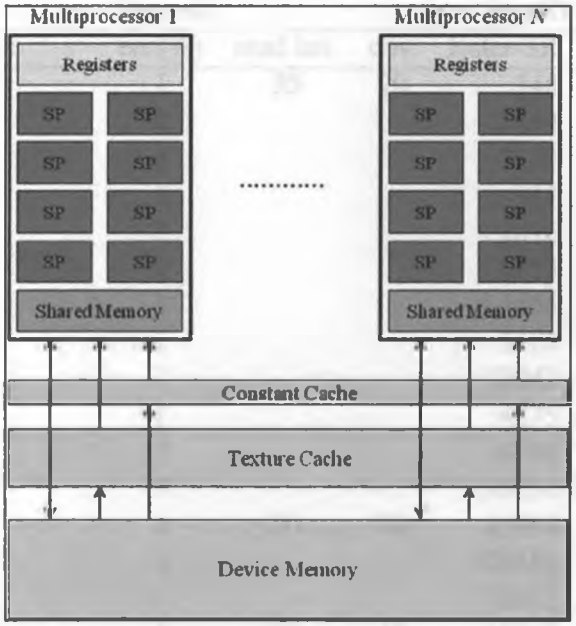


Figure 3.4: CUDA's hardware model [52].

3.3.2 Error correction

The error correction is done based on the spectral alignment algorithm and a Bloom filter data structure [9]. A Bloom filter represents a set of given keys in a bit-vector (Figure 3.5). Several hash functions are used for insertion and querying of keys. The spectrum of the set of reads is first constructed, represented by a Bloom filter. The spectrum is transformed into the CUDA texture memory. Then, they run the spectral alignment algorithm for correcting the errors in parallel.

They compared their method on a set of simulated benchmarks of length 35 with coverage 70. The comparison of running time of Euler-SR and CUDA implementation of EULER-SR is

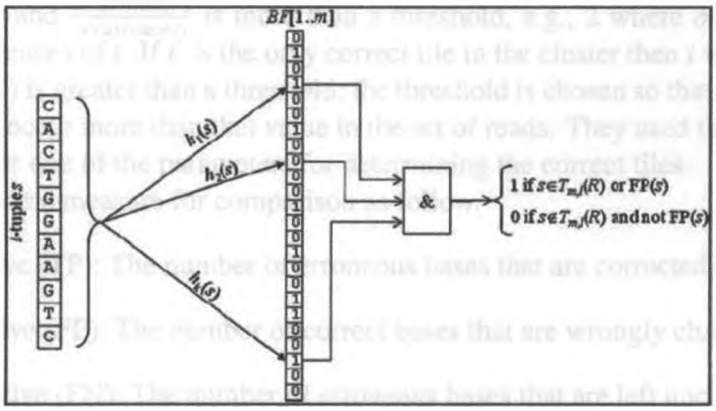


Figure 3.5: Bloom filter data structure. [52]

	Data set			Run Time(s)	
Genome	err.(%)	read len.	cov.	Euler-SR	CUDA
S.cer 5	1	35	70	345	26
	2			1454	70
	3			3239	92
S.cer 7	1			911	47
	2			2876	105
	3			6153	174
H.inf	1			1455	82
	2			5195	187
	3			12042	315
E.col	1			4875	279
	2			14793	669
	3			30250	1153
H.inf	1	70	70	15758	229
	2			578458	607
	3			70312	832
S.aureus	1	35	43	13260	230
H.acinonychis	1.6	36	190	7867	526

Table 3.3: Run time comparison between CUDA implementation and Euler-SR [52].

shown in Table 3.3. It can be concluded that CUDA is significantly faster than Euler-SR.

3.4 REPTILE

Yang et al. [65] proposed the newest method, Reptile, also based on k-mers. They attempt to correct erroneous k-mers based on contextual information as follow. A tile $t = a_1||a_2$ is a concatenation of overlapping k-mers a_1 and a_2. First clusters of tiles are created based on their Hamming distance and a threshold d. An erroneous tile t is corrected to a tile t' if the Hamming distance between t and t' is less than the Hamming distance between t and any other tile in the cluster and $\frac{occurrence(t')}{occurrence(t)}$ is more than a threshold, e.g., 2 where $occurrence(t)$ is the number of occurrences of t. If t' is the only correct tile in the cluster then t will be corrected to t' if $occurrence(t')$ is greater than a threshold; the threshold is chosen so that a large percentage (e.g. 4%) of tiles occur more than that value in the set of reads. They used the quality of reads, if it is available, as one of the parameters for determining the correct tiles.

They define some measure for comparison as follow.

- True Positive (TP): The number of erroneous bases that are corrected to a correct base.
- False Positive (FP): The number of correct bases that are wrongly changed.
- False Negative (FN): The number of erroneous bases that are left unchanged.
- True Negative (TN): The number of correct bases that are left unchanged.

Data set [£]	read len.	No. of Reads(M)	cov.	err.(%)
SRX000429[‡]	36	20.8	160	0.6
SRR001665_1[‡]	36	10.4	80	0.6
SRR006332[†]	36	17.7	173	1.5
D[§]	36	4.0	40	1.5
SRR022918_1[‡]	47	7.0	71	3.3
SRR034509_1[‡]	101	8.9	193	2.2

[£] Represented by accession number of read sets in NCBI.
[‡] Illumina reads from the E. coli str. K-12 substr (NC_000913) genome.
[†] Illumina reads from the Acinetobacter sp. ADP1 (NC_005966) genome.
[§] Set of reads generated randomly from SRR006332.

Table 3.4: The sets of real Illumina reads used in [65].

- n_e: The number of erroneous bases that are correctly identified but changed to a wrong base.

Sensitivity is the percentage of actual erroneous bases which are correctly corrected (Eq. 3.3). Specificity is the percentage of actual correct bases which are left unchanged (Eq. 3.4). Erroneous Base Assignment, EBA (Eq. 3.5), means how well erroneous bases are corrected to the true bases after a sequencing error has been identified. A lower value of EBA indicates a more accurate base assignment. Gain is the number of remaining erroneous bases divided by the number of actual erroneous bases (Eq. 3.6).

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.3}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.4}$$

$$\text{EBA} = \frac{n_e}{\text{TP} + n_e} \tag{3.5}$$

$$\text{Gain} = \frac{\text{TP} - \text{FP}}{\text{TP} + \text{FN}} \tag{3.6}$$

Table 3.4 gives the sets of real Illumina reads used for their comparison. As seen in the Table 3.5, Reptile outperforms SHREC in terms of gain, time, and space. Similar to all previous work, Reptile is required to tune several parameters in order to achieve high gain.

Genome	Method(d)	EBA(%)	Sensitivity(%)	Specificity(%)	Gain(%)	CPU Hours	Memory (GB)
SRX000429	SHREC	1.794	70.4	99.9	53.8	-	>8
	Reptile(1)	0.007	79	99.9	75.7	0.79	1.1
	Reptile(2)	0.028	86.4	99.9	80.2	2.49	1.1
SRR001665_1	SHREC	1.549	75.5	99.9	61.0	3.6	7.1
	Reptile(1)	0.009	67.8	99.9	65.2	0.35	0.84
	Reptile(2)	0.042	76.2	99.9	70.9	1.23	0.84
SRR006332	SHREC						
	Reptile(1)	0.013	75.1	99.8	63.2	1.66	2.2
D	SHREC	1.306	73.5	99.6	42.9	2.78	7.6
	Reptile(1)	0.091	71	99.8	59.9	0.26	0.66
SRR022918_1	SHREC						
	Reptile(1)	0.017	52.7	99.7	38.1	0.94	1.9
SRR034509_1	SHREC						
	Reptile(1)	0.01	85.3	99.9	78.9	2.76	4.6

Table 3.5: Comparison between Reptile and SHREC [65].

Chapter 4

A NEW APPROACH

4.1 Suffix Array and Longest Common Prefix Array

In this section, the basic definitions for strings and the suffix array data structure are described. Let s be a string over the alphabet Σ = {A, C, G, T}. A substring of s is any consecutive sequence of letters from s, i.e., $s[i \cdots j] = s[i]s[i+1] \cdots s[j]$. Suffix of s starting at ith position, $\texttt{suff}_i$, is a substring of s with $j = |s|$. Similarly, a prefix of s is a substring of s with $i = 1$. In order to get the reverse complement $\bar{s}$ of s, first the transformation A $\leftrightarrow$ T and C $\leftrightarrow$ G for all letters in s is applied. Then the resulting string is reversed. For example, if s = CAT, then $\bar{s}$ = ATG. It is obvious that $\bar{\bar{s}} = s$. The suffix array of s, denoted SA, gives the lexicographical order of the suffixes of s, i.e., $\texttt{suff}_{\texttt{SA}[1]} < \texttt{suff}_{\texttt{SA}[2]} < \cdots < \texttt{suff}_{\texttt{SA}[|s|]}$. In other words, $\texttt{SA}[i] = j$ if and only if $\texttt{suff}_j$ is the ith lexicographically smallest suffix of s. For example, the suffix array of the string "CATTATTAGGA" is shown in the second column of Table 4.1.

The suffix array is often used in combination with the longest common prefix (LCP) array that gives the length of the longest common prefix between consecutive suffixes of SA. Let $lcp(\alpha, \beta)$ denote the longest common prefix between strings α and β. Then, $\texttt{LCP}[i] = |lcp(\texttt{suff}_{\texttt{SA}[i-1]}, \texttt{suff}_{\texttt{SA}[i]}|$, is the length of the lcp between $\texttt{suff}_{\texttt{SA}[i-1]}$ and $\texttt{suff}_{\texttt{SA}[i]}$; see the fourth column of Table 4.1. By definition, LCP[1] = 0.

The suffix array data structure was introduced by [35]; SA can be computed in $O(m)$ time and space by any of the algorithms of [27, 29, 31]; the LCP array can be computed also in $O(m)$ time and space by the algorithm of [28]. However, suboptimal algorithms exist which behave much better in practice. We have used the libdivsufsort library of [38] in the implementation of our method. Also, since we need only bounded LCP values, we preferred a direct computation of the LCP, thus avoiding [28] altogether.

4.2 Basic idea for correcting

Consider a set of short reads $\{r_1, r_2, \cdots, r_n\}$ that have been sequenced from a genome $\mathcal{G}$ of length L. Each read r_i has a length l with per-base error rate p; $\mathcal{G}, r_i \in \Sigma^*$ = {A,C, G, T}* (Reads containing any letter not in Σ are discarded.)

Let read $r_i = xuay$ be sequenced from a position j of $\mathcal{G}$ where $x, u, y \in \Sigma^*$ and $a \in \Sigma$ and $|u| = w$, $|x| = k - w - 1$. Assume a is sequenced wrongly and $u = r_i[k - w \cdots k - 1]$ is correct.

i	SA	$\text{suff}_{\text{SA}[i]}$	LCP[i]
1	11	A	0
2	8	AGGA	1
3	5	ATTAGGA	1
4	2	ATTATTAGGA	4
5	1	CATTATTAGGA	0
6	10	GA	0
7	9	GGA	1
8	7	TAGGA	0
9	4	TATTAGGA	2
10	6	TTAGGA	2
11	3	TTATTAGGA	3

Table 4.1: Suffix array, SA, and LCP array of the string "CATTATTAGGA".

The letter $b = \mathcal{G}[j + k - 1]$ that actually appears in the genome is changed to the letter a in sequencing. Therefore, there are few occurrences of ua in $\mathcal{G}$. However, there are possibly other reads in R that are correctly sequenced around the same region as r_i, that is they contain the correct substring ub. The base a is suspected to be an error and it should actually be b because u is followed more often by b than by a. If there is a good coverage of reads, we can possibly correct a to the correct letter b.

A good data structure is required to analyze the huge amount of data generated by NGS in a feasible time. We use the suffix array which can be used as an index to quickly locate every occurrence of a substring within the string. First, we build the suffix array, SA, of all reads and their reverse complement, R

$$R = r_1\$\bar{r}_1\$r_2\$\bar{r}_2\$\cdots r_n\$\bar{r}_n\$,$$

where $\bar{r}_i$ is the reverse complement of read r_i and $\$ \notin \Sigma$. A witness u is any substring of R with a length of w. Support of $u \in \Sigma^*$ for $a \in \Sigma$ is defined as the number of occurrences of the substring ua in R, $\text{supp}(u, a)$.

In order to correct errors with high accuracy and sensitivity, a good estimation of parameters T and w are required. We estimated the parameters based on careful statistical techniques and provide the user with a full automatic tool for correcting the reads.

4.3 Statistical analysis

We now formalize the idea in the previous section. To correct errors in the reads, we go through the suffix array and cluster together positions that have a common prefix u of length w. The size of this cluster is

$$\texttt{clust}(u) = \sum_{a \in \Sigma} \text{supp}(u, a).$$

It is easy to compute the support values and cluster size, given the suffix array because all these positions are consecutive in SA and so are all occurrences of u supporting the same letter. The

Figure 4.1: An example of an error covered by six reads; the genome region where the reads came from is shown at the bottom. The letter (inside the frame) following the witness u = CTGTTGTCTC (underlined) should be T and not A. The support values are $\text{supp}(u, T) = 5$ and $\text{supp}(u, A) = 1$. If we omit the grey part, then the remaining suffixes are lexicographically sorted, as in SA.

clusters, corresponding to witnesses of a given length w, are easily found using the LCP values: a cluster consists of all consecutive positions with LCP values w or higher so that the $(w + 1)$st letter is not \$. In Figure 4.1, the occurrences of a witness are shown in the order in which they appear in the suffix array.

Assume for now that any witness u of length w does not appear elsewhere in the genome since additional occurrences would make the identification of the errors more difficult. However, it is not a precise assumption, because of the presence of repeats in the genome. The probability of random occurrences can be reduced in our Bernoulli model by adjusting the value of w. With a large value of w, the witness u is less likely to appear again in the genome but it will decrease its useful support because it will be covered by fewer reads. We are going to estimate the value of w using the statistical methods in Section 4.3.2.

4.3.1 Estimating the support values

Differentiating the correct and erroneous witnesses are required for estimating the support of a given witness u for a letter a following it. First, exact definitions are given since a witness may appear without errors in some reads and with errors in others. A witness is correct if it appears as a substring of the genome $\mathscr{G}$ and erroneous otherwise. Consider a witness $u = \mathscr{G}[i \cdots i + w - 1]$ followed a letter $a = \mathscr{G}[i + w]$. A read has to start within the interval $I = [[i - l + w + 1 \cdots i]$ to cover both u and a. Then, we estimate the support of u for a for each of the following cases.

1. Both u and a are correct.

2. u is correct but a is an error or u has one error but a is correct.

3. u has more than one error or both u and a are erroneous.

Case 1:
Denote by q_c the probability that a given read starts in the interval I and contains no errors

inside ua. In order to calculate q_c, we define two variables X and Y as follows:

$$X = \begin{cases} 1, & \text{if a given read starts inside the interval } I; \\ 0, & \text{otherwise.} \end{cases}$$

$$Y = \begin{cases} 1, & \text{if a given read containing } ua \text{ has no error inside } ua; \\ 0, & \text{otherwise.} \end{cases}$$

Each position has a Bernoulli distribution with $1 - p$ probability of success (correct base) and p probability of failure (erroneous base). With the assumption of independent occurrence of errors at each base, the probability that $w + 1$ consecutive bases are correct is $(1 - p)^{w+1}$. Therefore, q_c is calculated as shown in Eq. 4.1.

$$\begin{aligned} P(X = 1) &= \frac{l - w}{L}. \\ P(Y = 1) &= (1 - p)^{w+1}. \\ q_c &= P(X = 1 \text{ and } Y = 1) \\ &= P(X = 1) \cdot P(Y = 1) \\ &= \frac{l - w}{L}(1 - p)^{w+1}. \end{aligned} \tag{4.1}$$

Let R_c be the number of reads that start in the interval I and contain no errors inside ua.

$$P(R_c = k) = \binom{n}{k} {q_c}^k (1 - q_c)^{n-k}. \tag{4.2}$$

Then $W_c(k)$, the expected number of pairs (u, a) when both of them are correct, given $\text{supp}(u, a) = k$, is

$$\begin{aligned} W_c(k) &= \sum_{i=1}^{L} P(R_c = k) \\ &= \binom{n}{k} {q_c}^k (1 - q_c)^{n-k} L. \end{aligned} \tag{4.3}$$

Case 2:

Assume u is correct but a is an error. Let q_e be the probability that a given read covering ua with no error inside u but the original letter in a's position, say b, has been replaced by a.

$$Z = \begin{cases} 1 & \text{if a given read containing } ua \text{ has no error inside } u \text{ but an error in } a\text{'s position;} \\ 0 & \text{otherwise.} \end{cases}$$

a can be changed to either of the three other letters with probability p. Therefore, the probability that a is changed to one of them, say $b \neq a$, is $\frac{p}{3}$. Then, the probability that $P(Z = 1)$ is the probability of having no error inside u, $(1 - p)^w$, multiplied by the probability of having an error in a's position, $\frac{p}{3}$. Therefore,

$$\begin{aligned} P(Z = 1) &= \frac{p}{3}(1 - p)^w. \\ q_e &= P(X = 1 \text{ and } Z = 1) \\ &= P(X = 1) \cdot P(Z = 1) \\ &= \frac{l - w}{L}\frac{p}{3}(1 - p)^w. \end{aligned} \tag{4.4}$$

Let R_e be the number of reads that start in the interval I and contain no error inside u but an error in a's position. Then

$$P(R_e = k) = \binom{n}{k} {q_e}^k (1 - q_e)^{n-k}. \tag{4.5}$$

and $W_e(k)$, the expected number of pairs (u, a) when u is correct but a is an error, given $\text{supp}(u, a) = k$, is

$$\begin{aligned} W_e(k) &= \sum_{i=1}^{L} P(R_e = k) \\ &= \binom{n}{k} {q_e}^k (1 - q_e)^{n-k} L. \end{aligned} \tag{4.6}$$

The case when u has one error and a is correct is analogous.

Case 3

When u has more than one error or both u and a are erroneous, the support is much lower.

Now, we can compute a threshold, T, to differentiate the support by a correct witness for a correct letter from the support when either one or both of them are erroneous. We are required to find an interval for k such that $W_e(k)$ is smaller than $W_c(k)$. Any value in this interval is a good choice for T. Notice that with increasing the error rate this interval grows. Therefore the value of T remains good when the error rate reduces with correcting errors. Figure 4.2a shows the value of $W_c(k)$ and $W_e(k)$ for the genome of size 4.2 million with same number of reads and read length of 70 with per base error rate of 0.01. Figure 4.2b illustrates the region where both $W_c(k)$ and $W_e(k)$ are very small.

Such a region may not exist, for example in low coverage, when both $W_e(k)$ and $W_c(k)$ are very low. In order to cover also this case, the value of T is increased by an experimentally computed constant of two:

$$T = \min(\lfloor k | (W_c(k) > W_e(k)) \rfloor) + 2. \tag{4.7}$$

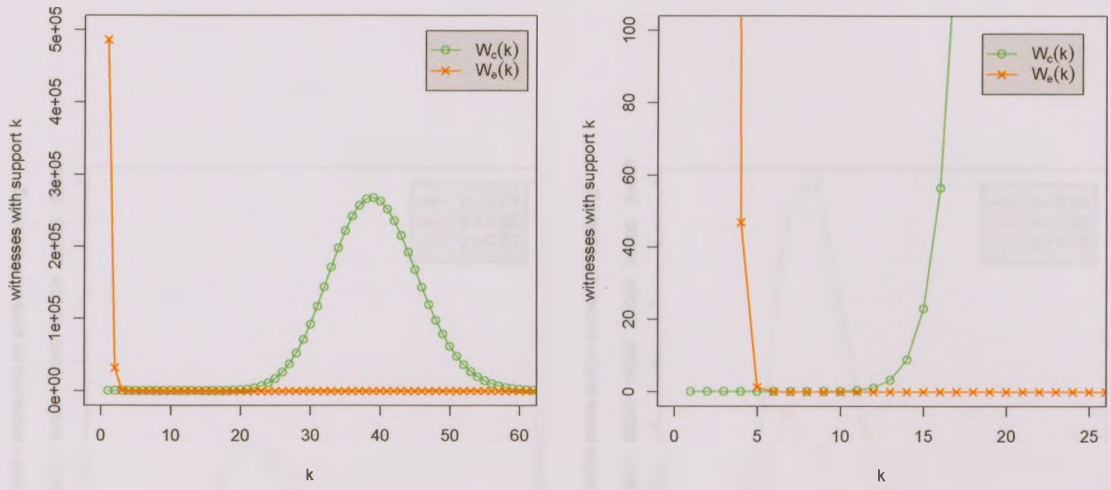
4.3.2 Estimating the witness length

We should also consider another case when errors in a read are distributed in such a way that no w consecutive correct positions exist. In this case, such reads can not be corrected using the current procedure because a correct witness at any position can not be fit. We are required to estimate a good value for w to reduce the chance of this. Let $f_w(k, l)$ be the number of possible ways to place k errors in a read of length l such that any interval of length w contains at least one error (Eq. 4.8).

$$f_w(k, l) = \begin{cases} \binom{l}{k}, & \text{if } l < w, \\ 0 & \text{if } k < \lfloor \frac{l}{w} \rfloor, \\ \sum_{i=1}^{w} f_w(k-1, l-i), & \text{otherwise.} \end{cases} \tag{4.8}$$

The probability of having k errors in a read of length l is,

$$f_w(k, l) p^k (1 - p)^{l-k}. \tag{4.9}$$



(a) (b)

Figure 4.2: a) The values of $W_c(k)$ and $W_e(k)$ for $L = n = 4.2$ mil., $l = 70$, $w = 21$, $p = 0.01$ b) the region of the (a) where the values of both $W_c(k)$ and $W_e(k)$ are very low. The value of the threshold T in this example equals 9.

Then the expected number of such reads is,

$$\sum_{i=1}^{n} f_w(k,l)p^k(1-p)^{l-k} = f_w(k,l)p^k(1-p)^{l-k}n. \tag{4.10}$$

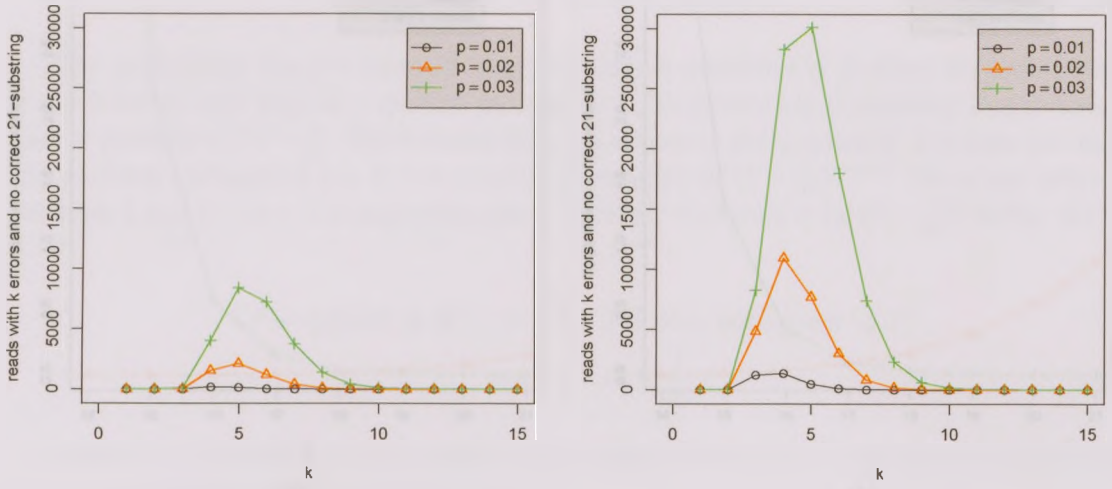
An example is illustrated in Figure 4.3 for a genome length of 4.2 billion and read length of 70 with different per base error rate. It can be obtained that the number of reads with k errors decreases with k but $f_w(k,l)$ increases and so the maximum is reached somewhere around 4-5 errors.

Therefore, the total number of reads uncorrectable with a witness of length w for different value of k is,

$$U(w) = \sum_{k=1}^{l} f_w(k,l)p^k(1-p)^{l-k}n. \tag{4.11}$$

Let the expected number of erroneous reads be $E_e = (1-(1-p)^l)n$. The percentage $U(w)$ represents out of E_e for $w = 21$ and $w = 18$ for different error rate is shown in Table 4.2. It can be seen from Figure 4.4 and Table 4.2 that decreasing the witness length will drop the number of uncorrectable reads.

However by decreasing the witness length we may face a new problem. The probability of the witness occurring more than once in the genome with small witness length increases. In this case, correct positions may be wrongly changed as follows. Consider the case that a is correct but its witness u is sampled as v in some reads ($|v| = |u|$). Assume that the probability



(a) $w = 21$ (b) $w = 18$

Figure 4.3: The number of reads with a given number of errors and no error-free interval of length w for $L = n = 4.2$ mil. and $l = 70$.

p	E_e	$U(w)/E_e(\%)$	
		$w = 21$	$w = 18$
0.01	2121678	0.15	0.02
0.02	3178885	0.87	0.17
0.03	3701953	2.56	0.68

Table 4.2: The percentage of the total number of reads uncorrectable with a witness of length w ($U(w)$) with a witness of length w out of the total number of expected erroneous reads (E_e).

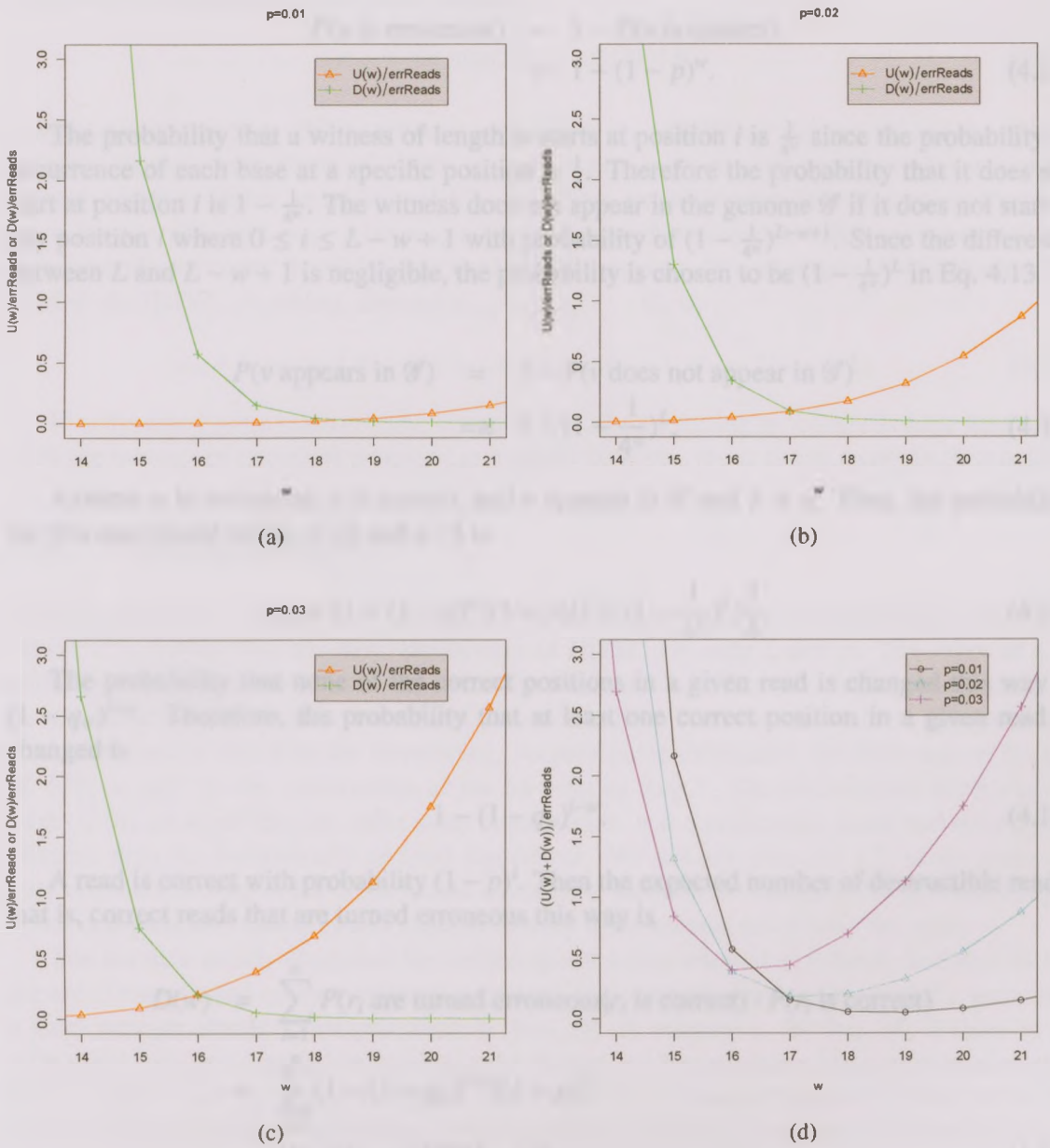


Figure 4.4: (a-b-c) The values of $U(w)$ and $D(w)$ as percentages of the total number of erroneous reads, E_e, for $L = n = 4.2$ mil., $l = 70$, and $p = 0.01$. (d) The values of $U(w) + D(w)$ as percentages of the total number of erroneous reads, E_e, for $L = n = 4.2$ mil., $l = 70$.

of v appearing in $\mathscr{G}$ is non-negligible, that is, v occurs in $\mathscr{G}$ at some position followed by $b \neq a$. Then supp(v, b) will be very large where supp(v, a) will be very small. In this case, a probably will be changed, incorrectly, into b. Therefore, we require to take into account such of errors.

The probability that u is erroneous is

$$\begin{aligned} P(u \text{ is erroneous}) &= 1 - P(u \text{ is correct}) \\ &= 1 - (1-p)^w. \end{aligned} \tag{4.12}$$

The probability that a witness of length w starts at position i is $\frac{1}{4^w}$ since the probability of occurrence of each base at a specific position is $\frac{1}{4}$. Therefore the probability that it does not start at position i is $1 - \frac{1}{4^w}$. The witness does not appear in the genome $\mathscr{G}$ if it does not start at any position i where $0 \leq i \leq L - w + 1$ with probability of $(1 - \frac{1}{4^w})^{L-w+1}$. Since the difference between L and $L - w + 1$ is negligible, the probability is chosen to be $(1 - \frac{1}{4^w})^L$ in Eq. 4.13.

$$\begin{aligned} P(v \text{ appears in } \mathscr{G}) &= 1 - P(v \text{ does not appear in } \mathscr{G}) \\ &=\cong 1 - (1 - \frac{1}{4^w})^L. \end{aligned} \tag{4.13}$$

Assume u is erroneous, a is correct, and v appears in $\mathscr{G}$ and $b \neq a$. Then, the probability for this case based on Eq. 4.12 and 4.13 is

$$q_w = (1 - (1-p)^w)(1-p)(1 - (1 - \frac{1}{4^w})^L)\frac{3}{4}. \tag{4.14}$$

The probability that none of the correct positions in a given read is changed this way is $(1 - q_w)^{l-w}$. Therefore, the probability that at least one correct position in a given read is changed is

$$1 - (1 - q_w)^{l-w}. \tag{4.15}$$

A read is correct with probability $(1-p)^l$. Then the expected number of destructible reads, that is, correct reads that are turned erroneous this way is

$$\begin{aligned} D(w) &= \sum_{i=1}^{n} P(r_i \text{ are turned erroneous} | r_i \text{ is correct}) \cdot P(r_i \text{ is correct}) \\ &= \sum_{i=1}^{n} (1 - (1 - q_w)^{l-w})(1-p)^l \\ &= (1 - (1 - q_w)^{l-w})(1-p)^l n. \end{aligned} \tag{4.16}$$

It can be seen from Figure 4.4 that lowering the witness length w decreases the number $U(w)$ of uncorrectable reads but increases the number $D(w)$ of destructible. Therefore a better evaluating function for choosing the witness length is to minimize $U(w) + D(w)$.

$$w_m = \arg\min_w (U(w) + D(w)). \tag{4.17}$$

It can be seen from Figure 4.4d that the optimal values for $p = 0.01, 0.02, 0.03$ are $w_m = 19$, 17, 16, respectively.

The highest accuracy for the current iteration and a greedy strategy can be obtained with $w = w_m$, theoretically. In order to avoid changing correct reads, a combination of values that are close to the optimal w_m and the smallest w,

$$w_M = \arg\min_w (D(w) < 0.0001 E_e) \tag{4.18}$$

works best in practice. While witnesses of length w_M effectively correct all but the uncorrectable $U(w_M)$ reads, those of length w_m will create large enough stretches of consecutive correct positions inside an additional $U(w_M) - U(w_m)$ reads so that they become correctable by witnesses of length w_M. Also, w_M satisfies the conditions under which we computed the parameter T and hence it will be also used for this purpose. The sequence of witness lengths used in the HiTEC algorithm, denoted $w_{seq} = w_{seq}[1 \cdots 9]$, is:

$$w_m + 1, w_M + 1, w_M + 1, w_m, w_M, w_M, w_m - 1, w_M - 1, w_M - 1. \tag{4.19}$$

Finally, since some reads contain several errors, the correcting procedure is done iteratively until the number of corrected positions in a single iterations drops below a certain threshold.

4.4 The algorithm

The pseudo code of the HiTEC algorithm that results from the above reasoning in shown is Figure 4.5. Notice that the input parameters of HiTEC are only L and p. The value of L is either known before the experiment or can be estimated by either a biological experiment or an expectation maximization procedure (such as in [40]). The value of p can be approximated from the machine that does the sequencing. As previously mentioned, the libdivsufsort library of [38] is used for the construction of the SA array in Step 7. The libdivsufsort library is the state-of-the-art algorithm for suffix array construction. It is significantly faster and more space efficient than the theoretically optimal algorithms. We did not store the LCP array since we only require it in Step 8, for construction of clusters. Instead, we calculate the LCP values by a direct computation. Cache effects ensured that the time remains essentially the same.

The iterative greedy algorithm for correcting erroneous letters is as follows. In Steps 10-11, the set of correct and erroneous letters supported by a witness u are constructed if u generates a large enough cluster. This procedure is done for all witness u. In Step 14, if there is no ambiguity, only one correct letter exists in the cluster and the erroneous letters will be corrected to the correct letter. In case of ambiguity, there is more than one correct letter in the cluster, the next two letters are checked in (Step 18) to choose the reliable correct letter. The position of a in the string R corresponds to a position inside a read r which can be some r_j or $\bar{r}_j$. In either case, we correct both r and its reverse complement $\bar{r}$.

The iterative procedure continues until the number of changed based in each iteration is less than 0.01% of the total number of bases (Step 22) or 9 iterations are performed. With increasing number of iterations, the ratio between the number of changed bases in one iteration and the total number of bases become less reliable as indicator of the actual number of corrected reads. Therefore, we add the last stopping condition.

We did not mention one practical improvement in the algorithm. In case of high coverage, HiTEC can split the data set into several sets of lower coverage. The algorithm in Figure 4.5 will be applied on each subset independently. This will decrease the space usage.

HiTEC $(r_1, r_2, \cdots r_n)$

- given: n reads $r_1 \cdots r_n$ (of length l each); L and p
- output: n corrected reads

```
1  compute w_m and w_M                                  //using Eq. 4.17 and 4.18, resp.
2  compute T                                            //using Eq. 4.7 with w = w_M
3  i ← 1                                                //iteration number
4  repeat
5      c ← 0                                            //bases changed this iteration
6      w ← w_seq[i]                                     //from Eq. 4.19
7      construct R and compute SA and LCP
8      compute the clusters in SA for all witnesses of length w
9      for all witness u with clust(u) ≥ T + 1
10         Corr ← {a | supp(u, a) ≥ T}
11         Err ← {a | supp(u, a) ≤ T − 1}
12         for all a ∈ Err
13             if (|Corr| = 1)
14                 correct a to b ∈ Corr                // change both r and r̄
15                 c ← c + 1
16             if (|Corr| ≥ 2)
17                 for all b ∈ Corr
18                     if (ua, ub followed by same two letters)
19                         correct a to b               // change both r and r̄
20                         c ← c + 1
21     i ← i + 1
22 untill ((c/ln < 0.0001) or (i > 9))
23 return all r_j s from R
```

Figure 4.5: The HiTEC algorithm.

Chapter 5

EXPERIMENTS

5.1 Accuracy

The ratio between the number of corrected reads and the number of initially erroneous reads is called accuracy. A read is correct if it appears as a substring in the genome and erroneous otherwise. A suffix array of the genome is built in order to search reads in the genome. Then the erroneous/correct status of all reads (before and after correction) has been found. We have:

- True Positive(TP): the number of erroneous reads that are corrected
- True Negative(TN): the number of correct reads that are left unchanged
- False Positive(FP): the number of correct reads that are wrongly changed
- False Negative(FN): the number of erroneous reads that are left unchanged
- err_{bef}: the number of erroneous reads before correction ($TP + FN$)
- err_{aft}: the number of erroneous reads after correction ($FP + FN$)

Then accuracy is defined in Eq. 5.1.

$$\begin{aligned} \text{accuracy} &= \frac{\text{err}_{\text{bef}} - \text{err}_{\text{aft}}}{\text{err}_{\text{bef}}} \\ &= \frac{TP - FP}{TP + FN} \end{aligned} \tag{5.1}$$

We have compared the accuracy of HiTEC with that of SHREC, CUDA and Reptile on a number of data sets (Table 3.1), including those of [50, 52, 65]. Several bacterial genomes, see Table 3.1, were downloaded from GenBank under the accession numbers given. We refer to these genomes by their IDs in parentheses in the first column of Table 3.1. We generated simulated data sets as those used in [50, 52] from the above genomes by uniformly sampling reads with a given length, coverage, and per-base error rate. That is, we do not use the same data sets but they were generated in the same way. According to the results, the performance of the programs does not depend on the generation of the data sets and therefore we can assume

Data set				Accuracy				
Genome	read len.	covrg.	err.(%)	SHREC	SHRECpaper	CUDA	Reptile	HiTEC
S.cer5	70	70	1	95.85	95.70		92.89	99.79
S.cer5	70	70	2	88.93	90.50		83.22	99.55
S.cer5	70	70	3	78.15	84.00		71.71	99.40
S.cer7	70	70	1	94.83	95.30		92.93	99.74
S.cer7	70	70	2	85.60	90.00		83.38	99.58
S.cer7	70	70	3	71.61	83.30		71.90	99.39
H.inf	70	70	1	91.21	94.10	87.50	93.00	99.73
H.inf	70	70	2	76.35	88.20	76.60	83.57	99.50
H.inf	70	70	3	55.84	81.00	63.60	72.08	99.02
E.coli	70	70	1	89.37	93.50		92.98	99.75
E.coli	70	70	2	71.38	87.40		83.45	99.42
E.coli	70	70	3	47.80	80.00		71.97	99.22

Table 5.1: Accuracy comparison for the data sets of [50]

our simulated data sets are identical with those of [50, 52]. All programs are run with their default parameters.

The comparison of the algorithms using the data sets used in [50] is shown in Table 5.1, All of data sets are generated by uniformly sampling reads of length 70 with coverage 70 for different per base error rate, 1, 2, and 3 %. The "SHREC" column represents the accuracy values we obtained by running the SHREC program whereas the values in the SHRECpaper column are taken from [50]. Since we were not able to run the "CUDA" implementation we just put the results in the paper [52] in the "CUDA" column. Thereby, some values are missing for these data sets since in [52] those tests have not been performed. It can be seen that HiTEC has accuracy over 99% for all data sets which is significantly higher than all previous results.

Table 5.2 shows the data sets used in [52] with smaller read length 35 and coverage 70. Since the results for SHREC provided in [50] did not include these data sets, we only put a column "SHREC" representing the accuracy values we obtained by running SHREC with its default parameters. In the "CUDA" column, accuracies from [52] is reported. Again HiTEC's accuracy for all data sets is over 90% and higher than both of SHREC and CUDA.

We also evaluate the proposed method on a mixture of read lengths and coverage levels taken from the longest genome, E.coli. These data sets are shown in Table 5.3 . The comparison was done by running the SHREC program and HiTEC. It can be seen that HiTEC with automatic tuning parameters has a stable performance for all different kinds of data sets with different read lengths and coverages. The accuracy of HiTEC is again over 90% and much higher than SHREC.

In order to be more precise, we did more comparison on several real sets of Illumina reads which are shown in Table 5.4. The first one, S.aureus, was also used in [50] and is available from *www.genomic.ch/edena.php*. This data set was previously used in [21]. Both the first and the second real data sets, S.aureus and H.acinonychis, were used in [52]. The second one, H.acinonychis, is available from *sharcgs.molgen.mpg.de/download.shtml* and it was used initially by [17]. The third one, E.coli2, is new and is available from clcbio.com/index.php?id=1290,

Data set				Accuracy		
Genome	read len.	covrg.	err.(%)	SHREC	CUDA	HiTEC
S.cer5	35	70	1	96.09	83.50	96.27
S.cer5	35	70	2	93.43	77.20	96.90
S.cer5	35	70	3	89.46	69.90	93.95
S.cer7	35	70	1	95.31	83.60	95.76
S.cer7	35	70	2	92.27	77.20	95.86
S.cer7	35	70	3	88.13	69.90	93.48
H.inf	35	70	1	93.34	83.50	96.39
H.inf	35	70	2	89.45	77.20	94.80
H.inf	35	70	3	83.93	69.90	89.83
E.coli	35	70	1	91.50	83.60	94.41
E.coli	35	70	2	87.06	77.20	94.37
E.coli	35	70	3	80.76	69.90	91.13

Table 5.2: Accuracy comparison for the data sets used in [52]

Data set				Accuracy	
Genome	read len.	covrg.	err.(%)	SHREC	HiTEC
E.coli	70	35	1	93.44	99.75
E.coli	70	35	2	87.87	99.46
E.coli	70	35	3	80.84	99.25
E.coli	50	50	1	93.44	99.25
E.coli	50	50	2	88.85	98.75
E.coli	50	50	3	83.65	97.88
E.coli	50	35	1	93.31	99.27
E.coli	50	35	2	89.20	99.06
E.coli	50	35	3	83.85	97.91
E.coli	35	50	1	91.60	94.37
E.coli	35	50	2	87.40	94.37
E.coli	35	50	3	82.32	91.15

Table 5.3: Accuracy comparison between SHREC and HiTEC for a variety of read lengths, coverage levels, and error rates sampled from the E.coli genome.

Genome	genome len.	read len.	original data set		after mapping	
			reads	coverage	reads	coverage
S.aureus	2,820,462	35	3,835,036	47.6	3,422,582	42.5
H.acinonychis	1,553,927	36	11,628,154	269.4	8,148,208	188.8
E.coli2	4,686,137	35	2,601,425	19.4	2,377,936	17.8

Table 5.4: List of several real sets of Illumina reads (original data set and after mapping using RMAP).

Data set				Accuracy			
Genome	read len.	covrg.	err.(%)	SHREC	SHRECpaper	CUDA	HiTEC
S.aureus	35	42.5	1.00‡	74.75	88.30	48.30	93.38
H.acinonychis	36	188.8	1.60†	34.83		47.40	91.26
E.coli2	35	17.8	0.38†	80.65			90.40

‡ Calculated from Eq. 5.4.

† Calculated from Eq. 5.3.

Table 5.5: Accuracy comparison for several real sets of Illumina reads listed in the "after mapping" column of Table 5.4.

the CLCbio web site, as an example of NGS data. A fourth real data set, E.coli, has been suggested by one of the reviewers of our paper [23] as a more recent example of Illumina reads with accession number ERA000206.

The real data sets were used in [50, 52] with a different coverage and number of reads (first and second data sets in Table 5.5). The comparison between the original data sets and the reduced ones after mapping are given in the fourth and fifth columns of Table 5.4. According to [48], the data sets were reduced by retaining only reads that mapped with three or less mismatches by mapping reads using RMAP, a read mapping software, [54]. The number of reads and the coverage after mapping are given in the last two columns of Table 5.4.

$$\text{./rmap -c chromosome dir -w 36 -m 3 -v -o output file reads file} \tag{5.2}$$

The per-base error rate is calculated by counting the total number of mismatches from the output file of RMAP. For instance, the number of reads that were mapped with $0, 1, 2$, and 3 mismatches for the S.aureus data set were 2,573,004, 589,619, 189,094, and 76,104, respectively. Therefore, the total number of mismatches is $1,196,119$. The per-base error rate (err) is

$$\begin{aligned} \text{err} &= \frac{\text{No. of mismatches}}{\text{total No. of bases}} && (5.3) \\ &= \frac{1,196,119}{35 \times 3,422,582} \\ &= .009985101 \approx 1\%. && (5.4) \end{aligned}$$

We compared the accuracy of SHREC, CUDA, and HiTEC on the reduced data sets and the results are shown in Table 5.5.

We have also tested SHREC and HiTEC on the original data sets and the results are shown in Table 5.6. In the case of the original data sets we could not map all reads and therefore could not provide the error rate in that way. We estimated the error rate by searching the reads in the genome and counting the number of erroneous reads, denoted by *errs*. We then estimated the error rate by applying a binomial distribution, it is given in Eq. 5.5. For example in the forth

Data set				Accuracy	
Genome7	read len.	covrg.	err.(%)	SHREC	HiTEC
S.aureus	35	47.6	1.06‡	57.47	74.21
H.acinonychis	36	269.4	2.06‡	15.23	53.91
E.coli2	35	19.4	0.41‡	59.26	68.20
E.coli	100	574	0.50†		87.49

‡ Calculated from Eq. 5.5.

† Calculated from Eq. 5.6.

Table 5.6: Accuracy comparison for several real sets of Illumina reads listed in the "original data set" column of Table 5.4.

Data set				Accuracy	
Accession number	read len.	covrg.	err.(%)	Reptile	HiTEC
SRX00429	36	160	0.44	84.32	86.17
SRR001665_1	36	80	0.38	75.28	85.78

Table 5.7: Accuracy comparison between Reptile and HiTEC on two sets of reads from the E.coli genome that were used in [65].

data set there are 10444830 erroneous reads among 26633604 total reads.

$$\begin{aligned} err &= 1-(1-\frac{errs}{n})^{\frac{1}{7}} && (5.5)\\ &= 1-((1-\frac{10444830}{26633604})^{\frac{1}{100}}) \\ &= 0.004966184 \approx 0.5\%. && (5.6) \end{aligned}$$

For the first three data sets, the accuracy of both programs is lower, as expected, but the advantage of HiTEC increases. SHREC program was not able to produce any results on the forth data set because of its very large size. The performance of HiTEC is very high.

Table 5.7 lists the relevant data sets from [65]. All of them are available in the Sequence Read Archive (SRA, *http://www.ncbi.nlm.nih.gov/sra*). However, for the data sets in SRA, the sequence of the genome from which the reads were sequenced is usually not known. (Indeed, there is no need to sequence a known genome, unless it is done precisely to provide data for verious tools.) When the genome is not known, the closest one known can be used but the results are not relevant since a very large proportion of the reads cannot be mapped.

The percentage of the reads that can be mapped (Table 5.8) is around 97% for the first two data sets in [65], indicated true reference genome, and only around 60-70% for the other ones. This is clear indication that the genomes for the last four data sets are not the actual genomes that produced the reads. Therefore, we just evaluated HiTEC on two of the data sets used in [65].

A discussion about all above results among all different experiments is given below. First, it can be obtained that HiTEC's accuracy is significantly higher than that of all the other programs for all experiments. It is also the case for the real data sets. While HiTEC's accuracy

Data	Allowed mismatches	Number of reads §	Uniquely mapped reads(%)	Ambiguously mapped reads(%)
SRX000429	5	20,708,709	96.5	2.5
SRR001665_1	5	10,359,952	96.7	2.5
SRR006332	5	17,675,271	79.9	1.5
D	5	4,000,000	84.1	1.6
SRR022918_1	10	7,049,153	62.5	1.5
SRR034509_1	10	8,874,761	63.5	1.2
	15		68.8	1.4

§ number of reads containing no ambiguous bases.

Table 5.8: Results of mapping each data sets in Table 3.4 to the corresponding genome using RMAP [65].

is not affected by the error rate or coverage, CUDAs accuracy is much lower compared to the simulated data and SHRECs accuracy decreases with the increase of the error rate. The accuracy of Reptile is clearly lower than that of HiTEC when quality scores are available (Table 5.7) and much lower when they are not (Table 5.1). HiTEC is also much less affected than Reptile by lower coverage.

Second, SHREC's accuracy is significantly different between the results from our tests of the software and those provided by [50], especially with increasing genome length and error rate. The difference appears because the best accuracy is obtained among a number of tuned parameters in [50]. As mentioned before, we run other programs with default parameters, without adjusting parameters. Also the accuracy values from [52] are higher than the values obtained by the tests of Euler-SR reported by [50]. The accuracy of HiTEC is significantly higher than all the other accuracy values, either the values reported in the published papers or the values from our experiments.

Third, in order to have a fair comparison between SHREC and HiTEC, we run SHREC program with the same number of iteration as ours, as resulted from the stopping criterion in Step 22. However, HiTEC's accuracy in most cases after one or two iterations is already higher that that of SHREC after nine iterations.

Similarly with SHREC, the parameters of Reptile are fixed. That means they do not adapt to the data. As a result, Reptile could not correct any errors of the fourth data set from Table 5.5.

Finally, we evaluate HiTEC using accuracy which is different from the measure, gain, used in [65]. The results are totally similar with our accuracy performance. For the data sets considered in Table 5.7, the gain for HiTEC is 83.33 and 82.22, respectively, whereas Reptile's is 82.81 and 72.53.

Data set				Time (s)			Space (MB)		
Genome	read len.	covrg.	err.(%)	SHREC	Reptile	HiTEC	SHREC	Reptile	HiTEC
S.cer5	70	70	1	3126	151	543	1340	727	399
S.cer5	70	70	2	4261	222	665	1350	531	399
S.cer5	70	70	3	7597	284	1193	1293	666	399
S.cer7	70	70	1	6511	373	1069	1502	1184	754
S.cer7	70	70	2	10649	618	1581	1512	1314	754
S.cer7	70	70	3	15823	792	2399	1745	1148	754
H.inf	70	70	1	9595	783	1971	1675	1434	1324
H.inf	70	70	2	15826	1340	2866	2090	1630	1324
H.inf	70	70	3	23319	1810	4253	3072	1771	1324
E.coli	70	70	1	23530	2741	5107	3194	1865	3210
E.coli	70	70	2	32073	5365	6290	3628	2266	3210
E.coli	70	70	3	57185	8812	11193	3437	2711	3210

Table 5.9: Time and space comparison between SHREC, Reptile and HiTEC.

5.2 Time and space comparison

We evaluated also the performance of all algorithms in terms of time and space (Table 5.9). The tests were performed for the data sets in Table 5.1 on a Sun Fire V440 Server, with four UltraSPARC IIIi processors at 1593MHz, 4GB RAM each, running SunOS 5.10. In addition to obtaining higher accuracy, HiTEC is time and space efficient, because of the use of good data structures. Our serial implementation of HiTEC is about six times faster than the multithreaded SHREC. The space required by our algorithm is comparable to that of Reptile and both are lower than SHRECs. Reptile is slightly faster however, the running time of HiTEC includes many iterations. In fact HiTEC may achieve higher accuracy sooner.

Chapter 6

CONCLUSION

Correcting errors in next generating sequencing data is highly demanded for further NGS applications. In spite of several programs for handling this issue, all of them require lots of parameters to be quite tuned. Beside tuning parameters, the proportion of the reads that are correctable using these methods is not very high. In this thesis we have provided an algorithm which is highly efficient at correcting the errors of next generating sequencing.

Our proposed method is based on the suffix array accompanied by statistical analysis. Because of huge amounts of data generated by these novel technologies, a good data structure is required to be able to analyze these data. The suffix array of a string can be used as an index to quickly locate every occurrence of a substring within the string. It can be constructed in linear time and space. Since reads are sampled several times randomly from different parts of genome with a high coverage, it can provide a good evidence to determine the erroneous bases. In order to correct errors with high accuracy and sensitivity, a good estimation of parameters are required. A careful statistical analysis is required to estimate those. We have provided the user with a fully automated tool for correcting the reads.

We have performed extensive comparisons with the best existing algorithms. They revealed that the accuracy of our algorithm is significantly higher than the accuracy of all previous algorithms. Our algorithm requires only the genome length and per-base error rate as the input parameters. Our algorithm is the only one which is able to automatically adjust to the input data.

We chose the data sets from Illumina reads. However, the approach can be applied to any type of reads for which the errors consist mainly of substitutions.

6.1 Very large genomes

We have also performed measurements to predict the ability of our algorithm to correct errors in the case of very large genomes. Figures 6.1 and 6.2 show the percentage of $U(w) + D(w)$ (uncorrectable plus destructible reads) for genome size 1GB with read lengths of 75 and 100, respectively, which are two common read lengths from Illumina.

In practice, the error rate increases with read length but so does our algorithms performance, only faster. While for reads of size 35 the ratio of those that can be corrected decreases below 50% for very large genomes, the situation is much better already for read size 50. In a 1GB

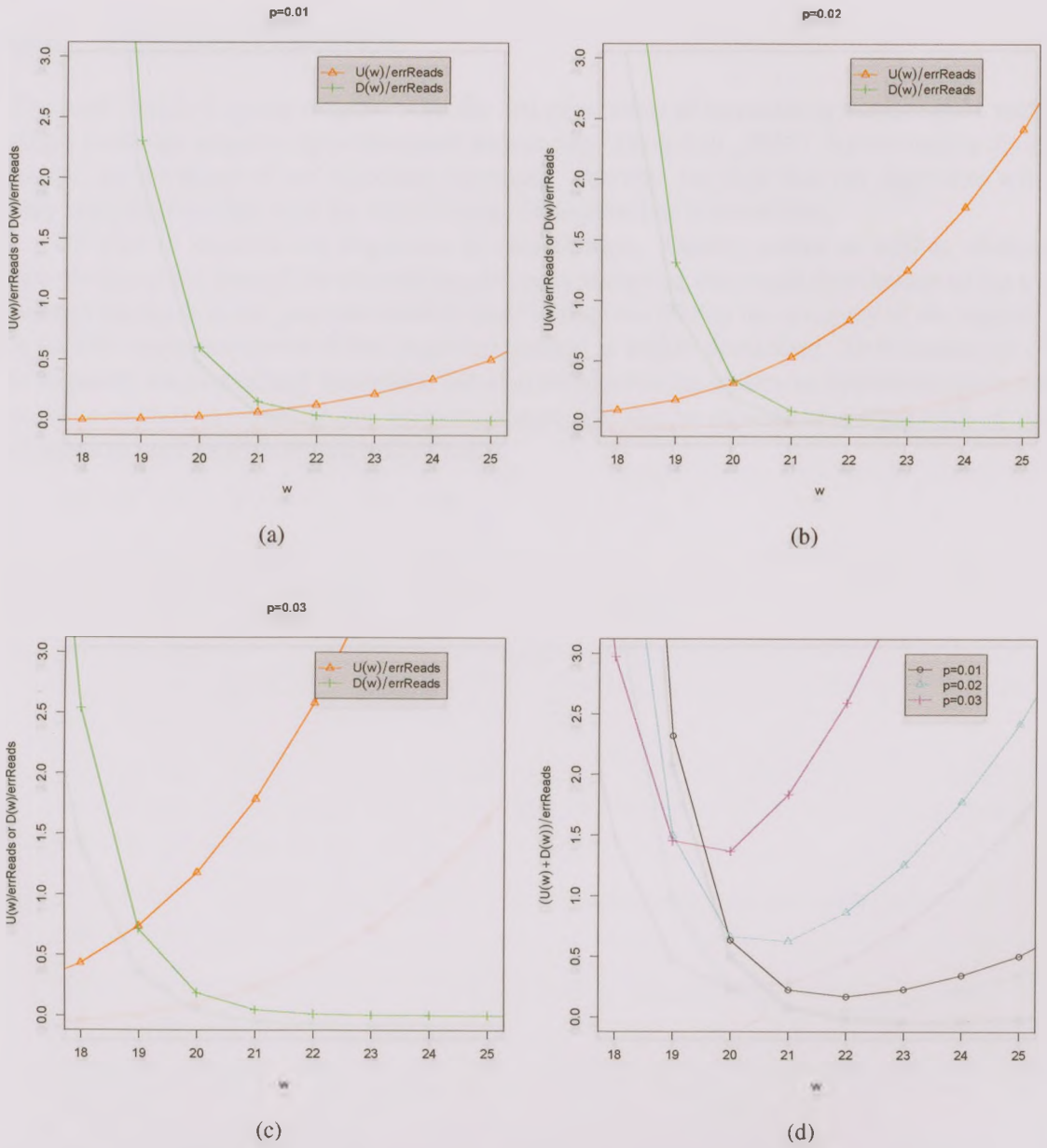


Figure 6.1: a-b-c) The values of $U(w)$ and $D(w)$ as percentages they represent out of the total number of erroneous reads, E_e, for $L = n = 1$ bil. and $l = 75$. d) The values of $U(w) + D(w)$ as percentages they represent out of the total number of erroneous reads, E_e, for $L = n = 1$ bil. and $l = 75$.

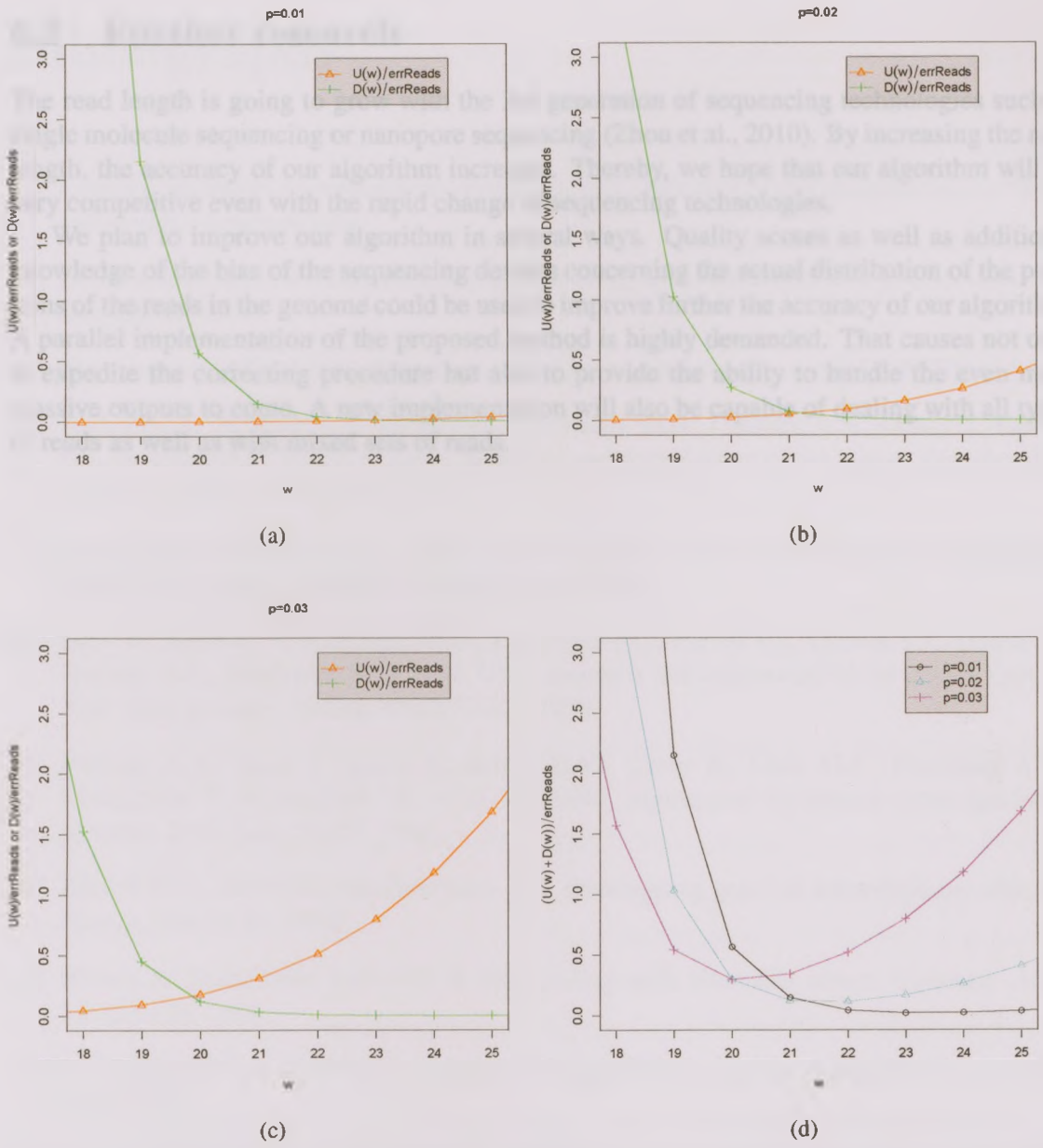


Figure 6.2: a-b-c) The values of $U(w)$ and $D(w)$ as percentages they represent out of the total number of erroneous reads, E_e, for $L = n = 1$ bil. and $l = 100$. d) The values of $U(w) + D(w)$ as percentages they represent out of the total number of erroneous reads, E_e, for $L = n = 1$ bil. and $l = 100$.

genome, for an error rate of 0.01 and a read length 50, our algorithm can correct up to 97.72% of the erroneous reads but when we increase the read length to 100, we can correct up to 97.70% even for a very high error rate of 0.03.

6.2 Further research

The read length is going to grow with the 3rd generation of sequencing technologies such as single molecule sequencing or nanopore sequencing (Zhou et al., 2010). By increasing the read length, the accuracy of our algorithm increases. Thereby, we hope that our algorithm will be very competitive even with the rapid change of sequencing technologies.

We plan to improve our algorithm in several ways. Quality scores as well as additional knowledge of the bias of the sequencing devices concerning the actual distribution of the positions of the reads in the genome could be used to improve further the accuracy of our algorithm. A parallel implementation of the proposed method is highly demanded. That causes not only to expedite the correcting procedure but also to provide the ability to handle the even more massive outputs to come. A new implementation will also be capable of dealing with all types of reads as well as with mixed sets of reads.

Bibliography

[1] [online]. available: *http://academic.brooklyn.cuny.edu/biology/bio4fv/page/molecular%20biology/dna-structure.html.*

[2] NCBI science primer. 2004. [online]. available: *http://www.ncbi.nlm.nih.gov/About/primer/images/Actg4.GIF.*

[3] The scientist. 2004. [online]. available: *http://www.the-scientist.com/2004/09/27/44/1/.*

[4] Anderson S., Bankier A.T., Barrell B.G., de Bruijn M.H., Coulson A.R., Drouin J., Eperon I.C., Nierlich D.P., Roe B.A., *et al.* Sequence and organization of the human mitochondrial genome. *Nature*, 290:457-465, 1981.

[5] Aluru S. Ko P. Lookup Tables, Suffix Trees and Suffix Arrays. Handbook of Computational Molecular Biology, Edited by Srinivas Aluru, 2005.

[6] Baer R., Bankier A.T., Biggin M.D., Deininger P.L., Farrell P.J., Gibson T.J., Hatfull G., Hudson G.S., Satchwell S.C., *et al.* DNA sequence and expression of the B95-8 Epstein-Barr virus genome. *Nature*, 310:207-211, 1984.

[7] Bankier A.T., Beck S., Bohni R., Brown C.M., Cerny R., Chee M.S., Hutchison C.A., Kouzarides T., Martignetti J.A., *et al.* The DNA sequence of the human cytomegalovirus genome. *DNA Seq.*, 2:1-12, 1991.

[8] Barrell B.G., Air G.M., and Hutchison C.A. Overlapping genes in bacteriophage phix174. *Nature*, 264:34-41, 1976.

[9] Bloom B. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13:422-426, 1970.

[10] Campagna D., *et al.* PASS: a program to align short sequences. *Bioinformatics*, 25:967-968, 2009.

[11] Campbell P.J., *et al.* Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. *Nat. Genet.* 40: 722-729, 2008.

[12] Chaisson M.J., Tang H., and Pevzner P.A. Fragment assembly with short reads. *Bioinformatics*, 20:2067-2074, 2004.

[13] Chaisson M.J. and Pevzner P.A. A short read fragment assembly of bacterial genomes. *Genome Res.*, 18:324-330, 2008.

[14] Chaisson M.J., *et al.* De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Res.*, 19:336-346, 2009.

[15] Cox-Foster D.L., *et al.* A metagenomic survey of microbes in honey bee colony collapse disorder. *Science* 318:283-287, 2007.

[16] Dahl F. *et al.* Multigene amplification and massively parallel sequencing for cancer mutation discovery. *Proc. Natl. Acad. Sci. USA* 104:9387-9392, 2007.

[17] Dohm J.C., *et al.* SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Res.*, 17:1697-1706, 2007.

[18] Eaves L.H. and Gao Y. MOM: maximum oligonucleotide mapping. *Bioinformatics*, 25:969-970, 2009.

[19] Fiers W., Contreras R., Haegemann G., Rogiers R., Van de Voorde A., Van Heuverswyn H., Van Herreweghe J., Volckaert G., and Ysebaert M. Complete nucleotide sequence of SV40 DNA. *Nature*, 273:113-120, 1978.

[20] Helmut Kae. The science creative quarterly. 2003. [online]. available: *http://www.scq.ubc.ca/genome-projects-uncovering-the-blueprints-of-biology/.*

[21] Hernandez D., *et al.* De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res.*, 18:802-809, 2008.

[22] Hutchison C.A. DNA sequencing: bench to bedside and beyond. *Nucleic Acids Research*, 35(18):6227-6237, 2007.

[23] Ilie L., Fazayeli F., and Ilie S. HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics*, 27(3), 295-302, 2011.

[24] Jeck W.R., *et al.* Extending assembly of short DNA sequences to handle error. *Bioinformatics*, 23:2942-2944, 2007.

[25] Jiang H. and Wong W.H. SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics*, 24:2395-2396, 2008.

[26] Johnson D.S., *et al.* Genome-wide mapping of in vivo protein-DNA interactions. *Science*, 316:1497-1502, 2007.

[27] *Kärkkäinen* J. and Sanders P. Simple linear work suffix array construction. *in Proc. of ICALP03, Lecture Notes in Comput. Sci. 2719, Springer-Verlag, Berlin, Heidelberg* 943-955, 2003.

[28] Kasai T., *et al.* Simple linear work suffix array construction. *Proc. of CPM01, Lecture Notes in Comput. Sci. 2089, Springer-Verlag, Berlin* 181192, 2001.

[29] Kim D.K., *et al.* Constructing suffix arrays in linear time. *J. Discrete Algorithms*, 3(2-4):126-142, 2005.

[30] Kim Y.J., *et al.* ProbeMatch: a tool for aligning oligonucleotide sequences. *Bioinformatics*, 25:1424-1425, 2009.

[31] Ko P. and Aluru S. Space efficient linear time construction of suffix arrays. *J. Discrete Algorithms*, 3(2-4):143-156, 2005.

[32] Lander E.S., *et al.* Initial sequencing and analysis of the human genome. *Nature*, 409:860-921, 2001.

[33] Langmead B., *et al.* Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, 10:R25, 2009.

[34] Lister R., *et al.* Highly integrated single-base resolution maps of the epigenome in Arabidopsis. *Cell* 133:523-536, 2008.

[35] Manber U. and Myers G. Suffix arrays: a new method for on-line search. *SIAM J. Comput.*, 22(5):935-948, 1993.

[36] Mardis E.R. The impact of next-generation sequencing technology on genetics. *Trends in Genetics*, 24(3):133-141, 2008.

[37] Metzker M.L. Sequencing technologies-the next generation. *Nature Reviews Genetics*, 11:31-46, 2010.

[38] Mori Y. libdivsufsort: A lightweight suffix sorting library. 2010. [online]. available: *http://code.google.com/p/libdivsufsort/.*

[39] Morin R.D., *et al.* Application of massively parallel sequencing to microRNA profiling and discovery in human embryonic stem cells. *Genome Res.*, 18:610-621, 2008.

[40] Myers G. Building fragment assembly string graphs. *Bioinformatics*, 21:ii79-ii85, 2005.

[41] Meyer M., Stenzel U. and Hofreiter M. Parallel tagged sequencing on the 454 platform. *Nat. Protocols* 3:267-278, 2008.

[42] Nickolls J., Buck I., Garland M., *et al.* Scalable parallel programming with cuda. *ACM Queue*, 6:39-55, 2008.

[43] Olena Morozova. Genome British Columbia. 2009. [online]. available: *http://www.genomebc.ca/education/articles/sequencing/.*

[44] Peer I. and Shamir R. Spectrum alignment: Efficient resequencing by hybridization. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (San Diego, CA)*, 260-268, 2000.

[45] Pevzner P.A. *et al.* An Eulerian path approach to dna fragment assembly. *Proc. Natl. Acad. Sci.*, 98:9748-9753, 2001.

[46] Sanger F., Nicklen S., and Coulson A.R. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci.*, 74:5463-5467, 1977.

[47] Sanger F., Coulson A.R., Hong G.F., Hill D.F. , and Petersen G.B. Nucleotide sequence of bacteriophage lambda DNA. *J. Mol. Biol.*, 162:729-773, 1982.

[48] Schmidt B. Personal communication. 2010.

[49] Schones D.E., *et al.* Dynamic regulation of nucleosome positioning in the human genome. *Cell* 132:887-898, 2008.

[50] Schroder J., *et al.* SHREC: a short-read error correction method. *Bioinformatics*, 25:2157-2163, 2009.

[51] Shendure J. and Ji H. Next-generation DNA sequencing. *Nat Biotech*, 26:1135-1145, 2008.

[52] Shi H., *et al.* A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. *J. Comput. Biol.*, 17:603-615, 2010.

[53] Simpson J.T., *et al.* ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19:1117-1123, 2009.

[54] Smith A.D., *et al.* Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC Bioinformatics*, 9:128, 2008.

[55] Smith L.M., Fung S., Hunkapiller M.W., *et al.* The synthesis of oligonucleotides containing an aliphatic amino group at the 5 terminus: synthesis of fluorescent DNA primers for use in DNA sequence analysis. *Nucleic Acids Res.*, 13:2399-2412, 1985.

[56] Smith M., Brown N.L., Air G.M., Barrell B.G., Coulson A.R., Hutchison C.A., and Sanger F. DNA sequence at the c termini of the overlapping genes A and B in bacteriophage phi X174. *Nature*, 265:702-705, 1977.

[57] Smith L.M., Fung S., Hunkapiller M.W., *et al.* The synthesis of oligonucleotides containing an aliphatic amino group at the 5 terminus: synthesis of fluorescent DNA primers for use in DNA sequence analysis. *Nucleic Acids Res.*, 13:2399-2412, 1985.

[58] Tammi M.T., *et al.* Correcting errors for shotgun sequencing. *J. Comput. Biol.*, 31:4663-4672, 2003.

[59] Tang H. Genome assembly, rearrangement, and repeats. *Chemical Reviews*, 107(8):3391-3406, 2007.

[60] Van Tassell C.P., *et al.* SNP discovery and allele frequency estimation by deep sequencing of reduced representation libraries. *Nat. Methods* 5:247-252, 2008.

[61] Venter J.C., Adams M.D., Myers E.W. The sequence of the human genome science. *Nature*, 291:1304-1351, 2001.

[62] Watson J.D. and Crick F.H.C. A structure for deoxyribose nucleic acid. *Nature*, 171:737-738, 1953.

[63] Wheeler D.A., *et al.* The complete genome of an individual by massively parallel DNA sequencing. *Nature*, 452:872-876, 2008.

[64] Wilhelm B.T., *et al.* Dynamic repertoire of a eukaryotic transcriptome surveyed at single-nucleotide resolution. *Nature* 453:1239-1243, 2008.

[65] Yang X., Dorman K.S., and Aluru S. Reptile: representative tiling for short read error correction. *J. Comput. Biol.*, 26:2526-2533, 2010.

[66] Zhou X.G., Ren L.F., Li Y.T., Zhang M., Yu Y.D., and Yu J. The next-generation sequencing technology: A technology review and future perspective. *SCIENCE CHINA Life Sciences*, 53:44-57, 2010.